UNIVERSITY OF CALIFORNIA SAN DIEGO

# Robot learning through Reinforcement Learning, Teleoperation and Scene Reconstruction

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Quan Ho Vuong

Committee in charge:

Professor Henrik Iskov Christensen, Chair
Professor Sergey Levine
Professor Hao Su
Professor Zhuowen Tu
Professor Michael Yip
Professor Rose Yu

2022

University of California San Diego

2022

## DEDICATION

To my family, friends and mentors.

TABLE OF CONTENTS

LIST OF FIGURES

# LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my PhD advisors Dr. Henrik Christensen and Dr. Hao Su. They provided the space I needed to develop as an independent researcher, introduced me to important technical areas and are always supportive when the needs arise. I would also like to thank my other committee members: Dr. Rose Yu, Dr. Zhuowen Tu Dr. Michael Yip and Dr. Sergey Levine. I also thank Dr. Keith Ross, my undergraduate mentor, for introducing me to Reinforcement Learning and working with me on my first Reinforcement Learning research paper.

I would like to thank my collaborators, from whom I have learned a tremendous amount, and in no particular order: Jiachen Li, Shuang Liu, Minghua Liu, Kamil Ciosek, Keith Ross, Henrik Iskov Christensen, Hao Su, Aviral Kumar, Yevgen Chebotar, Yiming Zhang, Kamil Ciosek, Robert Loftin, Katja Hofmann, Yuzhe Qin, Runlin Guo, Xiaolong Wang, Sharad Vikram, Sicun Gao, Sergey Levine.

I also thank the Government of Singapore and New York University Abu Dhabi for scholarships that enables my education.

Most importantly, I thank my family and parents for their continued support and encouragement.

Chapter 2, in part, is a reprint of material from Jiachen Li, Quan Vuong, Shuang Liu, Minghua Liu, Kamil Ciosek, Keith Ross, Henrik Iskov Christensen, Hao Su. “Multi-task Batch Reinforcement Learning with Metric Learning”. In Advances in Neural Information Processing Systems, 2020.

Chapter 3, in part, is a reprint of material from Quan Vuong, Aviral Kumar, Sergey Levine, Yevgen Chebotar. “Dual Generator Offline Reinforcement Learning”. In International Conference on Machine Learning Decision Awareness in Reinforcement Learning 2022 and In submission at Neural Information Processing Systems, 2022.

Chapter 4, in part, is a reprint of material from Quan Vuong, Yiming Zhang, and Keith Ross. “Supervised policy update for deep reinforcement learning”. In International Conference on Learning Representation, 2019.

Chapter 5, in part, is a reprint of material from Kamil Ciosek, Quan Vuong, Robert Loftin and Katja Hofmann. “Better Exploration with Optimistic Actor-Critic”. In Conference on Neural Information Processing Systems, 2019.

Chapter 6, in part, is a reprint of material from Quan Vuong, Yuzhe Qin, Runlin Guo, Xiaolong Wang, Hao Su, Henrik Christensen. “Single RGB-D Camera Teleoperation for General Robotic Manipulation”.

Chapter 7, in part, is a reprint of material from Quan Vuong, Sharad Vikram, Hao Su, Sicun Gao, Henrik I. Christensen. “How to pick the domain randomization parameters for sim-to-real transfer of reinforcement learning policies?”. In IEEE International Conference on Robotics and Automation Learning Legged Locomotion Workshop, 2019.

Chapter 8, in part, is a reprint of material from Hao Su, Henrik Christensen, Luobin Wang, Quan Vuong, Runlin Guo, Yuzhe Qin. Ongoing work in the research topic of geometric real2sim2real. Authors listed in alphabetical order. The dissertation author is the primary investigator.

## VITA

| | |
|---|---|
| 2018 | B. S. in Mathematics, New York University |
| 2022 | Ph. D. in Computer Science, University of California San Diego |

## PUBLICATIONS

Yiming Zhang, **Quan Vuong**, Kenny Song, Xiao-Yue Gong, Keith Ross. Efficient Entropy For Policy Gradient with Multi-Dimensional Action Space. In International Conference on Learning Representation Workshop, 2018.

**Quan Vuong**, Yiming Zhang, and Keith Ross. Supervised policy update for deep reinforcement learning. In International Conference on Learning Representation, 2019.

Kamil Ciosek, **Quan Vuong**, Robert Loftin, Katja Hofmann. Better Exploration with Optimistic Actor Critic. In Advances in Neural Information Processing Systems, 2019.

**Quan Vuong**, Sharad Vikram, Hao Su, Sicun Gao, Henrik Iskov Christensen. How to pick the domain randomization parameters for sim-to-real transfer of reinforcement learning policies? In IEEE International Conference on Robotics and Automation Learning Legged Locomotion Workshop, 2019.

Yiming Zhang, **Quan Vuong**, and Keith Ross. First Order Optimization in Policy Space for Constrained Deep Reinforcement Learning. In Advances in Neural Information Processing Systems, 2020.

Jiachen Li, **Quan Vuong**, Shuang Liu, Minghua Liu, Kamil Ciosek, Keith Ross, Henrik Iskov Christensen, Hao Su. Multi-task Batch Reinforcement Learning with Metric Learning. In Advances in Neural Information Processing Systems, 2020.

Che Wang, Yanqiu Wu, **Quan Vuong**, Keith Ross. Striving for Simplicity and Performance in Off-Policy DRL: Output Normalization and Non-Uniform Sampling. In International Conference on Learning Representation, 2020.

**Quan Vuong**, Yuzhe Qin, Runlin Guo, Xiaolong Wang, Hao Su, Henrik Iskov Christensen. Single RGB-D Camera Teleoperation for General Robotic Manipulation. Preprint.

**Quan Vuong**, Aviral Kumar, Sergey Levine, Yevgen Chebotar. Dual Generator Offline Reinforcement Learning. In submission at Neural Information Processing Systems, 2022.

ABSTRACT OF THE DISSERTATION

**Robot learning through Reinforcement Learning, Teleoperation and Scene Reconstruction**

by

Quan Ho Vuong

Doctor of Philosophy in Computer Science

University of California San Diego, 2022

Professor Henrik Iskov Christensen, Chair

Designing agents that autonomously acquire skills to complete tasks in their environments has been an ongoing research topic for decades. The complete realization of the vision remains elusive, yet research pursued in the quest toward this goal has yielded tremendous scientific and technological advances. The thesis addresses three research areas that are key to progress on this vision.

The first area is deep Reinforcement Learning (RL), where we develop new algorithms for both online and offline RL. More specifically, we propose an experimental setting where we demonstrate that pre-training policies from offline datasets can lead to significant improvement in online learning sample efficiency on unseen tasks (up to 80% on standard benchmarks). The second contribution in this area is a novel offline RL algorithm based on Generator Adversarial Network. In contrast to recent algorithms that enforce distribution constraints, we use a dual generator formulation to enforce support constraints, leading to improved performance. The method outperforms recent state-of-the-art algorithms on tasks that require stitching sub-optimal trajectories to learn performant behavior.

The second is human-machine interfaces for human supervision, e.g. to collect demonstrations for robotic manipulation. Using a single RGB-D camera as the sensing device to capture human motion in real-time, we demonstrate that our teleoperation system allows the human operator to successfully control a 6 degree-of-freedom manipulator to complete complex tasks, such as peg-in-hole and folding cloth.

The third is the automatic construction of simulated environments for training deep neural networks. We show the benefit of our framework in the task of grasping objects in clutter using 6 degree-of-freedom grasp. Using only 30 reconstructed scenes and thousands of grasp labels, a state-of-the-art grasping network architecture when trained using our reconstructions outperforms by 11% the publicly released pre-trained model that was trained with 17.7 million grasp labels.

# Chapter 1

# Introduction

Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain. – Alan Turing

In the same chapter where the famous quote above was written, Alan Turing offers a principle from which such a child machine can be educated: ”We normally associate punishments and rewards with the teaching process. Some simple child machines can be constructed or programmed on this sort of principle. The machine has to be so constructed that events which shortly preceded the occurrence of a punishment-signal are unlikely to be repeated, whereas a reward-signal increased the probability of repetition of the events which led up to it” [TUR50]

He went on to lament that ”I have done some experiments with one such child-machine, and succeeded in teaching it a few things, but the teaching method was too unorthodox for the experiment to be considered really successful.” He would be surely relieved to learn that 70 years later, reinforcement learning algorithms have been demonstrated to be general-purpose algorithms that can learn from punishments and reward signals to perform complex tasks.

The basic framework of Reinforcement Learning is as follows. The learning agent interacts with an environment. The learning agent receives as input observation and outputs an action, after which the agent receives a scalar reward signal indicating how desirable the action

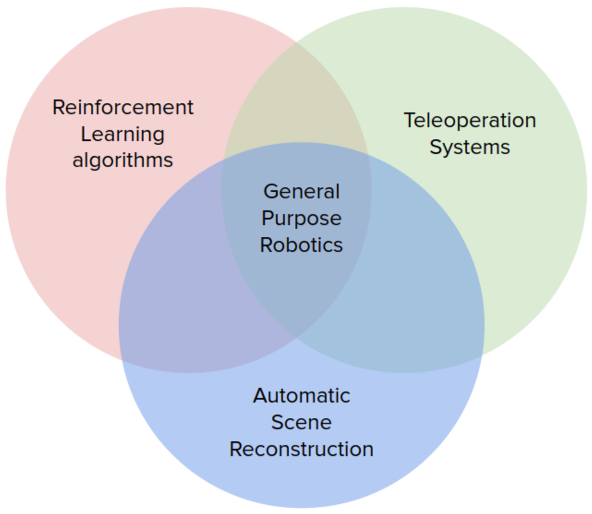

Reinforcement Learning algorithms
Teleoperation Systems
General Purpose Robotics
Automatic Scene Reconstruction

**Figure 1.1**: An illustrations of three research areas that are key ingredients for continued progress towards general purpose robotics.

was. A *successful* learning agent must use the reward signal to infer the optimal action that leads to the highest possible reward. The development of Reinforcement Learning can be traced back to the work by Richard Bellman on dynamic programming [Bel57] and temporal difference learning by Richard Sutton [SB18]. Early attempts at developing Reinforcement Learning algorithms that can solve complex real-world problems were hindered by the lack of expressive representation. They resorted to table look-up or linear function, neither of which were adequate [Tes95].

However, in recent years, deep neural networks have been proven to be an expressive class of representation, and when trained with Reinforcement Learning algorithms, can represent complex behaviors and solve practical real-world problems. A particularly intriguing property of the approach is its generality, having been successfully applied to problems from seemingly unrelated domains, such as video games [MKS+15a], robot grasping [KIP+18] and computer chip design [MGY+21, RRK+21]. Such generality brings a tremendous amount of excitement because the approach might open the door to previously unsolved scientific and technological problems.

However, the complete realization of Turing's original vision in designing agents that can be educated using reward and punishment signals remain elusive. In this thesis, we will present our research on three research areas that are key to continued progress towards this vision. Figure 1.1 provides an illustration of the three research areas. We next briefly describe the motivation that connects these three research areas.

Reinforcement Learning can be thought of as an algorithmic framework to perform task-oriented end-to-end learning. This is appealing because the algorithm can train deep neural networks to perform the task of interest without the designer having to explicitly engineer representation and reasoning capability into the network. The appropriate representation and reasoning capability can instead emerge out of the learning procedure. In this case, it is interesting to note the similarity with behavior-based robotics, which has been argued to lead to intelligent behavior without explicitly engineered representation [Bro91b] or reasoning system [Bro91a]. Because of this benefit, our first area of research interest is in designing better *Reinforcement Learning algorithms*. However, a fundamental challenge that Reinforcement Learning algorithms face is the so-called bootstrapping problem, where learning can be prohibitively slow at the beginning due to the lack of demonstration of the desired behavior. A possible method to circumvent this issue is to collect human demonstrations, which is a popular practice when applying Reinforcement Learning to robotic manipulation. The importance of human demonstration motivates our second area of interest, which is in developing new *teleoperation systems*. Last but not least, another appealing aspect of Reinforcement Learning is that it has the potential to train agents to perform many tasks in a scalable manner. To do so, we would need an environment where the learning agents can propose their own tasks to practice, without the algorithm designer having to specify what those tasks are a priori. Creating such an environment in the real world is costly and time-consuming, and therefore we turn our attention to simulation. More concretely, the third research area that we are interested in is the *automatic reconstruction* of real-world scenes for training neural networks.

In the sections below, we describe the contribution in each research area in more details.

## 1.1 Algorithms for offline Reinforcement Learning

The successes of deep neural networks in recent years hinge on training them on large-scale and diverse datasets. In contrast, Reinforcement Learning has been traditionally framed as an *online* decision-making problem, where the learning agent interleaves collecting fresh experiences and learning from them. While such formulation encapsulates the two main steps that any intelligent agents perhaps should possess, the need to collect new experiences before learning from them means that the dataset size is often limited, especially in domains where collecting data is either expensive or unsafe.

The limitation of *online* Reinforcement Learning therefore has ignited interest in *offline* Reinforcement Learning in recent years. In offline Reinforcement Learning [LGR12], sometimes known as batch Reinforcement Learning, the learning agent must learn from a static dataset and is not allowed to interact with the environment during the learning process. After learning finishes, the learning agent is then evaluated in the task of interest, either with or without adapting to new experiences. If we can develop stable and performant offline Reinforcement Learning algorithms and train them on large-scale offline datasets, perhaps we will see Reinforcement Learning agents enjoying the same level of generalization and usefulness that deep networks trained with supervised learning currently enjoy.

Our first work in this area demonstrates the promise of offline Reinforcement Learning when trained from diverse datasets. In this work, given multiple datasets collected from different tasks, we train a multi-task policy to perform well in unseen tasks sampled from the same distribution. Given the network trained using the offline datasets, when we allow further training on the unseen tasks, using the trained policy as an initialization leads to significantly faster convergence compared to randomly initialized policies (up to 80% improvement and across 5 different Mujoco task distributions). We name our method Multi-task Batch RL with Metric Learning. We discuss this work in detail in chapter 2.

Offline Reinforcement Learning also faces a unique set of challenges, one of which is constraining the learned policy to remain close to the data. Such a constraint is essential to prevent the policy from outputting out-of-distribution (OOD) actions with erroneously overestimated values. In principle, generative adversarial networks (GAN) can provide an elegant solution to do so, with the discriminator directly providing a probability that quantifies distributional shift. However, in practice, GAN-based offline RL methods have not outperformed alternative approaches, perhaps because the generator is trained to both fools the discriminator and maximize return - two objectives that are often at odds with each other. In this paper, we show that the issue of conflicting objectives can be resolved by training two generators: one that maximizes return, with the other capturing the ”remainder” of the data distribution in the offline dataset, such that the mixture of the two is close to the behavior policy. We show that not only does having two generators enable an effective GAN-based offline RL method, but also approximates a support constraint, where the policy does not need to match the entire data distribution but only the slice of the data that leads to high long-term performance. We name our method DASCO, for Dual-Generator Adversarial Support Constrained Offline RL. On benchmark tasks that require learning from sub-optimal data, DASCO significantly outperforms prior methods that enforce distribution constraints. We discuss this work in detail in chapter 3.

## 1.2 Algorithms for online Reinforcement Learning

In spite of the discussions regarding the limitation of *online* Reinforcement Learning in the previous subsection, developing efficient *online* Reinforcement Learning remains an open, interesting and impactful intellectual endeavor. For example, in Offline Reinforcement Learning research, we are often interested in using online learning to further improve the performance of the policy trained with offline data. We will discuss two of our contributions to online Reinforcement Learning. The first contribution is a theoretically motivated algorithm called *Supervised Policy*

*Update*. In online Reinforcement Learning, given the current policy and a fixed number of recently collected trajectories, we need to search for a new policy before collecting more experience. It is often desirable to limit this search to a space of policies close to the current policy to ensure learning stability. Our algorithm proposes to solve this search by decomposing it into two steps: solving for the non-parameterized policy, and then parameterizing the non-parameterized policy with a neural network. Interestingly, the theoretical formulation of the problem suggests a simple early stopping condition when training using a new batch of experience. The algorithm is applicable to both discrete and continuous action spaces. We discuss *Supervised Policy Update* in chapter 4.

The second contribution that we include in this thesis is an algorithm for better exploration techniques for actor-critic algorithms. Exploration techniques are crucial to improve the sample efficiency of online Reinforcement Learning algorithms because exploration directly determines the quality of data that the learning algorithm is exposed to. In this work, we proposed *Optimistic Actor Critic*, which explores more efficiently by operationalizing the principle of optimism in the face of uncertainty to actor-critic algorithms. We discuss the algorithms in more detail in chapter 5.

## 1.3 Teleoperation system

In previous sub-sections, we describe our contributions in the form of new algorithms for online and offline Reinforcement Learning. These algorithms were tested on standard simulated benchmarks, which is a helpful step toward demonstrating their usefulness and allowing others to reproduce the reported results. However, one of the most promising applications of Reinforcement Learning is to teach robots to perform tasks in the real world, especially tasks that involve manipulating objects. This is because the real world is unstructured, and dynamic, making it challenging to develop accurate physical models. Since Reinforcement Learning algorithms do not require plausible physical models, applying Reinforcement Learning to robotics, therefore, holds the promise that robots can autonomously acquire useful skills without needing to have accurate physical models.

Unsurprisingly, applying learning methods to robotic manipulation has its own sets of challenges, one of which is how to *bootstrap* the learning process. Before learning, a learning agent might not have any useful knowledge of the world, or possesses any skills, it is therefore unlikely that the agent can produce sequences of interactions that will complete the task of interest. In such a case, the learning agent can not make any progress. A common method to bootstrap the learning process is to provide the learning agent with successful demonstrations of the task. Given the demonstrations, the agent can acquire a baseline level of mastery and therefore obtain positive rewards, which allow the agent to reinforce behaviors to produce positive rewards and therefore improve over time.

There are many different methods to provide learning agents with demonstrations to perform manipulation tasks. While a review of these different methods is beyond the scope of the thesis, we refer interested readers to [SK07] for a comprehensive discussion. In our work, we propose a teleoperation system that uses a single RGB-D camera as the human motion capture device. Our system can perform general manipulation tasks such as cloth folding, hammering, and 3mm clearance peg-in-hole. We propose the use of a non-Cartesian oblique coordinate frame, dynamic motion scaling, and repositioning of operator frames to increase the flexibility of our teleoperation system. Demos of our systems are available online at https://sites.google.com/view/manipulation-teleop-with-rgbd. We describe the details of our system in chapter 6.

## 1.4 Real2Sim2Real

Because of the data required in training deep neural networks, learning inside a simulated environment and then transferring the trained policies to the real world has become an influential and active research area. Several impressive demonstrations of Reinforcement Learning on real robots made extensive use of simulators for learning, including quadrupedal locomotion over challenging natural terrain [LHW$^+$20] and dextrous manipulation [ABC$^+$].

However, these approaches usually require practitioners to manually curate the object meshes, place them at realistic poses in the simulation scenes and calibrate their dynamics parameters. The process of manual scene creation and calibration requires domain expertise and can be prohibitively costly to scale to large-scale scenes with many objects. Perhaps this is one of the reasons why applications using Sim2Real have mostly been demonstrated on manipulation tasks in constrained settings involving a single object [XCB$^+$22], such as rope manipulation, or when the simulated scenes can be procedurally generated, such as in bin picking [MLN$^+$17, MPH$^+$16, MML$^+$17]. Recognizing scene creation and calibration as a major bottleneck of Sim2Real, recent research has attempted to automate this process and dub the problem *Real2Sim2Real* [LHC$^+$21].

We present two separate contributions to this research area. In chapter 7, we demonstrate that the dynamic parameter of the simulated scene can be optimized using RL to maximize the performance of the trained policies in the test environment. The performance of the policies trained with the simulation instance tuned using our method improves over the performance of reasonable default setting of the simulation instance by up to 50%. While our work in chapter 7 is concerned with tuning the dynamics parameter, in chapter 8, we instead focus on the geometric problem of automatically reconstructing the object meshes in a scene and placing them at realistic poses. We show the benefit of our framework in the task of grasping objects in clutter using 6 degree-of-freedom grasp. Using only 30 reconstructed scenes and thousands of grasp labels, a state-of-the-art grasping network architecture when trained using our reconstructions outperforms by 11% the publicly released pre-trained model that was trained with 17.7 million grasp labels.

# Chapter 2

# Multi-task Batch Reinforcement Learning with Metric Learning

Combining neural networks (NN) with reinforcement learning (RL) has led to many recent advances [WWVR20, SWD$^+$17, HZAL18a, SLM$^+$15, VZR18]. Since training NNs requires diverse datasets and collecting real world data is expensive, most RL successes are limited to scenarios where the data can be cheaply generated in a simulation. On the other hand, offline data is essentially free for many applications and RL methods should use it whenever possible. This is especially true because practical deployments of RL are bottle-necked by its poor sample efficiency. This insight has motivated a flurry of recent works in Batch RL [SSB$^+$20, ASN19, KFS$^+$19a, FMP19, CZW$^+$19]. These works introduce specialized algorithms to stabilize training from offline datasets. However, offline datasets are not necessarily diverse. In this work, we investigate how the properties of a diverse dataset influence the policy search procedure. By collecting diverse offline dataset, we hope the networks will generalize without further training to unseen tasks or provide good initialization that speeds up convergence when we perform further on-policy training.

To collect diverse datasets, it occurs to us that we should collect data from different tasks.

However, datasets collected from different tasks may have state-action distributions with large divergence. Such dataset bias presents a unique challenge in robust task inference. We provide a brief description of the problem setting, the challenge and our contributions below. For ease of exposition, we refer to such datasets as having little overlap in their state-action visitation frequencies thereafter.

We tackle the Multi-task Batch RL problem. We train a policy from multiple datasets, each generated by interaction with a different task. We measure the performance of the trained policy on unseen tasks sampled from the same task distributions as the training tasks. To perform well, the policy must first infer the identity of the unseen tasks from collected transitions and then take the appropriate actions to maximize returns. To train the policy to infer the task identity, we can train it to distinguish between the different training tasks when given transitions from the tasks as input. These transitions are referred to as the context set [RZQ$^+$19]. Ideally, the policy should model the dependency of the task identity on both the rewards and the state-action pairs in the context set. To achieve this, we can train a task identification network that maps the collected experiences, including both state-action pairs and rewards, to the task identity or some task embedding. This approach, however, tends to fail in practice. Since the training context sets do not overlap significantly in state-action visitation frequencies, it is possible that the learning procedure would minimize the loss function for task identification by *only* correlating the state-action pairs and ignoring rewards, which would cause mistakes in identifying testing tasks. This is an instance of the well-known phenomena of ML algorithms cheating when given the chance [CZS17] and is further illustrated in Fig. 2.1. We limit our explanations to the cases where the tasks differ in reward functions. Extending our approach to task distribution with different transition functions is easily done. We provide experimental results for both cases.

Our contributions are as follows. To the best of our knowledge, we are the first to highlight the issue of the task inference module learning the wrong correlation from biased dataset. We propose a novel application of the triplet loss to robustify task inference. To mine hard negative

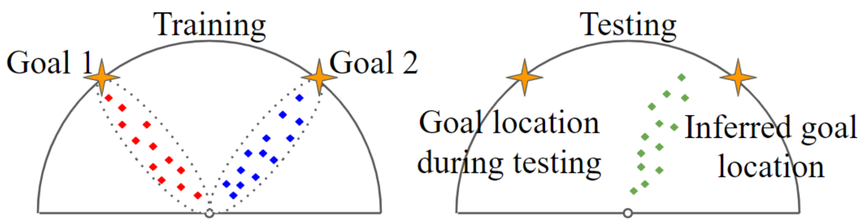


**Figure 2.1**: A toy example to illustrate the challenge. The agent must navigate from the origin to a goal location. **Left:** Goal 1 and Goal 2 denote the two training tasks. The red and blue squares indicate the transitions collected from task 1 and 2 respectively. We can train the task inference module to infer the task identity to be 1 when the context set contains the red transitions and 2 when the context set contains the blue transitions. Since there are no overlap between the red and blue squares, the task inference module learns to correlate the state-action pairs to the task identity. **Right:** The failure of the task inference module. The policy must infer the task identity from the randomly collected transitions, denoted by the green squares. The agent needs to navigate to goal 1 during testing. However, if the green squares have more overlap with the blue squares, the task inference module will predict 2 to be the task identity. The agent therefore navigates to the wrong goal location.

examples, we approximate the reward function of each task and relabel the rewards in the transitions from the other tasks. When we train the policy to differentiate between the original and relabelled transitions, we force it to consider the rewards since their state-action pairs are the same. Training with the triplet loss generalizes better to unseen tasks compared to alternatives. When we allow further training on the unseen tasks, using the policy trained from the offline datasets as initialization significantly increase convergence speed (up to 80% improvement in sample efficiency).

To the best of our knowledge, the most relevant related work is [SSB$^+$20], which is solving a different problem from ours. They assume access to the ground truth task identity and reward function of the testing task. Our policy does not know the testing task's identity and must infer it through collected trajectories. We also do not have access to the reward function of the testing tasks.

## 2.1 Preliminaries and Problem Statement

We model a task as a Markov Decision Process $M = (\mathcal{S}, \mathcal{A}, T, T_0, R, H)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $T$, initial state distribution $T_0$, reward function $R$, and horizon $H$. At each discrete timestep $t$, the agent is in a state $s_t$, picks an action $a_t$, arrives at $s'_t \sim T(\cdot|s_t, a_t)$, and receives a reward $R(s_t, a_t, s'_t)$. The performance measure of policy $\pi$ is the expected sum of rewards $J_M(\pi) = \mathbb{E}_{\tau_M \sim \pi}[\sum_{t=0}^{H-1} R(s_t, a_t, s'_t)]$, where $\tau_M = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$ is a trajectory generated by using $\pi$ to interact with $M$.

### 2.1.1 Batch Reinforcement Learning

A Batch RL algorithm solves the task using an existing batch of $N$ transitions $\mathcal{B} = \{(s_t, a_t, r_t, s'_t)|t = 1, \ldots, N\}$. A recent advance in this area is Batch Constrained Q-Learning (BCQ) [FMP19]. Here, we explain how BCQ selects actions. Given a state $s$, a generator $G$ outputs multiple candidate actions $\{a_m\}_m$. A perturbation model $\xi$ takes as input the state-candidate action and generates small correction $\xi(s, a_m)$. The corrected action with the highest estimated $Q$ value is selected as $\pi(s)$:

$$\pi(s) = \operatorname*{arg\,max}_{a_m + \xi(s, a_m)} Q(s, a_m + \xi(s, a_m)), \qquad \{a_m = G(s, \nu_m)\}_m, \qquad \nu_m \sim \mathcal{N}(0, 1). \tag{2.1}$$

In our paper, we use BCQ as a routine. The take-away is that BCQ takes as input a batch of transitions $\mathcal{B} = \{(s_t, a_t, r_t, s'_t)|t = 1, \ldots, N\}$ and outputs three learned functions $Q, G, \xi$.

### 2.1.2 Multi-task Batch Reinforcement Learning

Given $K$ batches, each containing $N$ transition tuples from one task, $\mathcal{B}_i = \{(s_{i,t}, a_{i,t}, r_{i,t}, s'_{i,t}) | i = 1, \ldots, K, t = 1, \ldots, N\}$, we define the Multi-task Batch RL problem as:

$$\arg\max_{\theta} J(\theta) = \mathbb{E}_{M_i \sim p(M)} \left[ J_{M_i}(\pi_\theta) \right], \tag{2.2}$$

where an algorithm only has access to the $K$ batches and $J_{M_i}(\pi)$ is the performance of the policy $\pi$ in task $i$, i.e. $\mathbb{E}_{\tau_{M_i} \sim \pi}[\sum_{t=0}^{H-1} R(s_{i,t}, a_{i,t}, s'_{i,t})]$. $p(M)$ defines a task distribution. The subscript $i$ indexes the different tasks. The tasks have the same state and action space and only differ in the transition and reward functions [ZSI$^+$20]. A distribution over the transition and/or the reward functions therefore defines the task distribution. We measure performance by computing average returns over unseen tasks sampled from the same task distribution. The policy is not given identity of the unseen tasks before evaluation and must infer it from collected transitions.

In multi-task RL, we can use a task inference module $q_\phi$ to infer the task identity from a context set. The context set for a task $i$ consists of transitions from task $i$ and is denoted $\mathbf{c}_i$. The task inference module $q_\phi$ takes $\mathbf{c}_i$ as input and outputs a posterior over the task identity. We sample a task identity $\mathbf{z}_i$ from the posterior and inputs it to the policy in addition to the state, i.e. $\pi(s, \mathbf{z}_i)$. We model $q_\phi$ with the probabilistic and permutation-invariant architecture from [RZQ$^+$19]. $q_\phi$ outputs the parameters of a diagonal Gaussian. For conciseness, we sometimes use the term policy to also refer to the task inference module. It should be clear from the context whether we are referring to $q_\phi$ or $\pi$.

We evaluate a policy on unseen tasks in two different scenarios: (1) Allowing the policy to collect a small number of interactions to infer $z$, we evaluate returns without further training, (2) Training the policy in the unseen task and collecting as much data as needed, we evaluate the amount of transitions the policy needs to collect to converge to the optimal performance.

We assume that each batch $\mathcal{B}_i$ contains data generated by a policy while learning to solve task $M_i$. Thus, if solving each task involve visiting different subspace of the state space, the different batches do not have significant overlap in their state-action visitation frequencies. This is illustrated in Fig. 2.1.

## 2.2 Proposed algorithm

### 2.2.1 Learning multi-task policy from offline data with distillation

In Multi-task RL, [RCG$^+$15, TBC$^+$17, GSR$^+$17, CPO$^+$19, PBS15] demonstrate the success of distilling multiple single-task policies into a multi-task policy. Inspired by these works, we propose a distillation procedure to obtain a multi-task policy in the Multi-task Batch RL setting. In Sec. 2.2.2, we argue such distillation procedure alone is insufficient due to the constraints the batch setting imposes on the policy search procedure.

The distillation procedure has two phases. In the first phase, we use BCQ to learn a different policy for each task, i.e. we learn $K$ different and independent policies. While we can use any Batch RL algorithm in the first phase, we use BCQ due to its simplicity. As described in Sec. 2.1.1, for each training batch, BCQ learns three functions: a state-action value function $Q$, a candidate action generator $G$ and a perturbation generator $\xi$. The output of the first phase thus consists of three sets of networks $\{Q_i\}_{i=1}^K$, $\{G_i\}_{i=1}^K$, and $\{\xi_i\}_{i=1}^K$, where $i$ indexes over the training tasks.

In the second phase, we distill each set into a network by incorporating a task inference module. The distilled function should recover different task-specific function depending on the inferred task identity. To distill the value functions $\{Q_i\}_{i=1}^K$ into a function $Q_D$, for each task $i$, we sample a context $\mathbf{c}_i$ and a pair $(s,a)$ from the batch $\mathcal{B}_i$. The task inference module $q_\phi$ takes $\mathbf{c}_i$ as input and infers a task identity $\mathbf{z}_i$. Given $\mathbf{z}_i$ as input, $Q_D$ should assign similar value to $(s,a)$ as the value function for the $i^{th}$ task $Q_i(s,a)$. The loss function with a $\beta$-weighted KL term [RZQ$^+$19] is:

$$\mathcal{L}_Q = \frac{1}{K}\sum_{i=1}^{K} \mathop{\mathbb{E}}_{(s,a),\mathbf{c}_i\sim\mathcal{B}_i} \left[(Q_i(s,a) - Q_D(s,a,\mathbf{z}_i))^2 + \beta\mathrm{KL}(q_\phi(\mathbf{c}_i)||\mathcal{N}(0,1))\right], \mathbf{z}_i \sim q_\phi(\mathbf{c}_i) \quad (2.3)$$

We also use Eq. 2.3 to train $q_\phi$ using the reparam trick [KW13]. Similarly, we distill the candidate action generators $\{G_i\}_{i=1}^K$ into $G_D$. $G_D$ takes as input state $s$, random noise $\nu$ and task identity $\mathbf{z}_i$. Depending on $\mathbf{z}_i$'s value, we train $G_D$ to regress towards the different candidate action generator:

$$\mathcal{L}_G = \frac{1}{K}\sum_{i=1}^{K} \mathop{\mathbb{E}}_{\substack{s,\mathbf{c}_i\sim\mathcal{B}_i \\ \nu\sim\mathcal{N}(0,1)}} \left[||G_i(s,\nu) - G_D(s,\nu,\bar{\mathbf{z}}_i)||^2\right], \quad \mathbf{z}_i \sim q_\phi(\mathbf{c}_i). \quad (2.4)$$

The bar on top of $\bar{\mathbf{z}}_i$ in Eq. 2.4 indicates the stop gradient operation. We thus do not use the gradient of Eq. 2.4 to train the task inference module [RZQ+19]. Lastly, we distill the perturbation generators $\{\xi_i\}_{i=1}^K$ into a single network $\xi_D$ (Eq. 2.5). $\xi_D$ takes as input a state $s$, a candidate action $a$, and an inferred task identity $\mathbf{z}_i$. We train $\xi_D$ to regress towards the output of $\xi_i$ given the same state $s$ and candidate action $a$ as input. We obtain the candidate action $a$ by passing $s$ through the candidate action generator $G_i$.

$$\mathcal{L}_\xi = \frac{1}{K}\sum_{i=1}^{K} \mathop{\mathbb{E}}_{\substack{s,\mathbf{c}_i\sim\mathcal{B}_i \\ \nu\sim\mathcal{N}(0,1)}} \left[||\xi_i(s,a) - \xi_D(s,a,\bar{\mathbf{z}}_i)||^2\right], \quad \mathbf{z}_i \sim q_\phi(\mathbf{c}_i), \quad a = G_i(s,\nu) \quad (2.5)$$

Note that the gradient of $\mathcal{L}_\xi$ also updates $G_i$. The final distillation loss is given in Eq. 2.6. We parameterize $q_\phi, Q_D, G_D, \xi_D$ with feedforward NN.

$$\mathcal{L}_{distill} = \mathcal{L}_Q + \mathcal{L}_G + \mathcal{L}_\xi. \quad (2.6)$$

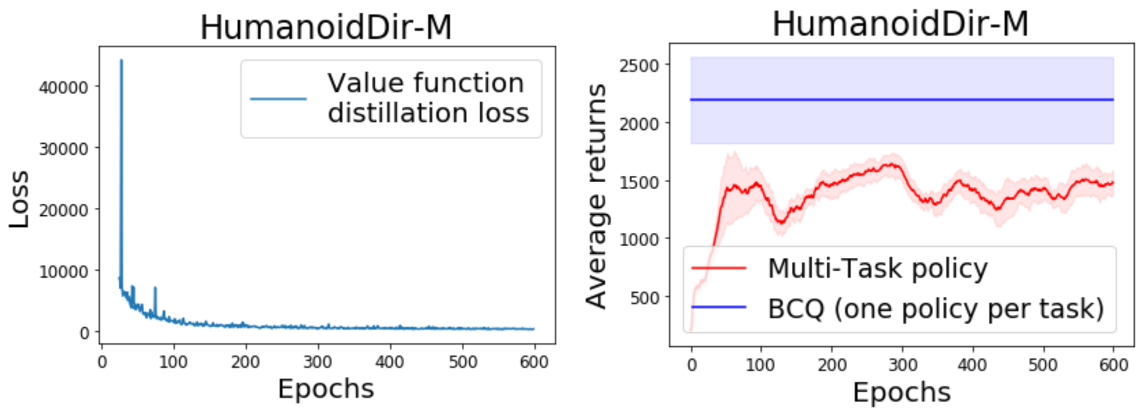


**Figure 2.2**: **Top:** Value function distillation loss (Eq. 2.3) during training. **Bottom:** The performance of the multi-task policy trained with Eq. 2.6 versus BCQ.

### 2.2.2 Robust task inference with triplet loss design

Given the high performance of distillation in Multi-task RL [RCG+15, TBC+17, GSR+17, CPO+19, PBS15], it surprisingly performs poorly in Multi-task Batch RL, even on the training tasks. This is even more surprising because we can minimize the distillation losses (Fig. 2.2 top) and the single-task BCQ policies have high performance (Fig. 2.2 bottom). If the single-task policies perform well and we can distill them into a multi-task policy, why does the multi-task policy have poor performance? We argue the task inference module has learnt to model the posterior over task identity as conditionally dependent on only the state-action pairs in the context set , i.e. $P(Z|S,A)$, where $S,A$ are random variables denoting states and actions, rather than the correct dependency $P(Z|S,A,R)$ where $R$ denotes the rewards.

The behavior of the trained multi-task policy supports this argument. In this experiment, each task corresponds to a running direction. To maximize returns, the policy should run with maximal velocity in the target direction. We found that the multi-task policy often runs in the wrong target direction, indicating incorrect task inference. At the beginning of evaluation, the task identity is not provided. The policy takes random actions, after which it uses the collected transitions to infer the task identity. Having learnt the wrong conditional dependency, the task inference module assigns high probability mass in the posterior to region in the task embedding space whose training batches overlap with the collected transitions (Fig. 2.1).

The fundamental reason behind the wrong dependency is the non-overlapping nature of the training batches. Minimizing the distillation loss does not require the policy to learn the correct but more complex dependency. The multi-task policy should imitate different single-task policy depending on which batch the context set was sampled from. If the batches do not overlap in state-action visitation frequencies, the multi-task policy can simply correlate the state-action pairs in the context with which single-task policy it should imitate. In short, if minimizing the training objective on the given datasets does not require the policy to model the dependency of the task identity on the rewards in the context set, there is no guarantee the policy will model this dependency. This is not surprising given literature on the non-identifiability of causality from observations [Pea09, PJS17]. They also emphasize the benefit of using distribution change as training signal to learn the correct causal relationship [BDR+20].

Inspired by this literature, we introduce a distribution change into our dataset by approximating the reward function of each task $i$ with a learned function $\hat{R}_i$. Given a context set $\mathbf{c}_j$ from task $j$, we relabel the reward of each transition in $\mathbf{c}_j$ using $\hat{R}_i$. Let $t$ index the transitions and $\mathbf{c}_{j\to i}$ denote the set of the relabelled transitions, we illustrate this process below :

$$\mathbf{c}_j = \left\{\left(s_{j,t}, a_{j,t}, r_{j,t}, s'_{j,t}\right)\right\}_t \xrightarrow{\text{Relabelling}} \mathbf{c}_{j\to i} = \left\{\left(s_{j,t}, a_{j,t}, \hat{R}_i(s_{j,t}, a_{j,t}), s'_{j,t}\right)\right\}_t \tag{2.7}$$

Given the relabelled transitions, we leverage the triplet loss from the metric learning community [HBL17] to enforce robust task inference, which is the most important design choice in MBML. Let $K$ be the number of training tasks, $\mathbf{c}_i$ be a context set for task $i$, $\mathbf{c}_j$ be a context set for task $j$ $(j \neq i)$ , and $\mathbf{c}_{j\to i}$ be the relabelled set as described above, the triplet loss for task $i$ is:

$$\mathcal{L}^i_{triplet} = \frac{1}{K-1} \sum_{j=1, j\neq i}^{K} \Bigg[ \underbrace{d\left(q_\phi\left(\mathbf{c}_{j\to i}\right), q_\phi\left(\mathbf{c}_i\right)\right)}_{\substack{\text{Ensure } \mathbf{c}_{j\to i} \text{ and } \mathbf{c}_i \text{ infer} \\ \textit{similar} \text{ task identities}}} - \underbrace{d\left(q_\phi\left(\mathbf{c}_{j\to i}\right), q_\phi\left(\mathbf{c}_j\right)\right)}_{\substack{\text{Ensure } \mathbf{c}_{j\to i} \text{ and } \mathbf{c}_j \text{ infer} \\ \textit{different} \text{ task identities}}} + \quad m \Bigg]_+ , \tag{2.8}$$

**Algorithm 1** Calculating the distillation and triplet loss

**Input**: Batches $\{\mathcal{B}_i\}_{i=1}^K$; BCQ-trained $\{Q_i\}_{i=1}^K$, $\{G_i\}_{i=1}^K$, and $\{\xi_i\}_{i=1}^K$; randomly initialized $Q_D$, $G_D$ and $\xi_D$ jointly parameterized by $\theta$; task inference module $q_\phi$ with randomly initialized $\phi$

```
1: repeat
2:     Sample context set $\mathbf{c}_i$ from $\mathcal{B}_i, \forall i$
3:     Obtain relabelled transitions $\mathbf{c}_{j\to i}$ according to Eq. 2.7 for all pair of task $i, j$
4:     Calculate $\mathcal{L}_{triplet}$ using Eq. 2.9
5:     Calculate $\mathcal{L}_Q, \mathcal{L}_G, \mathcal{L}_\xi$ using Eq. 2.3, 2.4, 2.5
6:     Calculate $\mathcal{L}$ using Eq. 2.10
7:     Update $\theta, \phi$ to minimize $\mathcal{L}$
8: until Done
```

where $m$ is the triplet margin, $[\cdot]_+$ is the ReLU function and $d$ is a divergence measure. $q_\phi$ outputs the posterior over task identity, we thus choose $d$ to be the KL divergence.

Minimizing Eq. 2.8 accomplishes two goals. It encourages the task inference module $q_\phi$ to infer similar task identities when given either $\mathbf{c}_i$ or $\mathbf{c}_{j\to i}$ as input. It also encourages $q_\phi$ to infer different task identities for $\mathbf{c}_j$ and $\mathbf{c}_{j\to i}$. We emphasize that the task inference module can not learn to correlate *only* the state-action pairs with the task identity since $\mathbf{c}_j$ and $\mathbf{c}_{j\to i}$ contain the same state-action pairs, but they correspond to different task identities. To minimize Eq. 2.8, the module must model the correct conditional dependency $P(Z|S,A,R)$ when inferring the task identity.

Eq. 2.8 calculates the triplet loss when we use the learned reward function of task $i$ to relabel transitions from the remaining tasks. Following similar procedures for the remaining tasks lead to the loss:

$$\mathcal{L}_{triplet} = \frac{1}{K}\sum_{i=1}^{K} \mathcal{L}_{triplet}^{i}. \tag{2.9}$$

The final loss to train the randomly initialized task inference module $q_\phi$, the distilled value functions $Q_D$, the distilled candidate action generator $G_D$, and the distilled perturbation generator $\xi_D$ is:

$$\mathcal{L} = \mathcal{L}_{triplet} + \mathcal{L}_Q + \mathcal{L}_G + \mathcal{L}_\xi. \tag{2.10}$$

Alg. 1 illustrates the pseudo-code for the second phase of the distillation procedure. In theory, we can also use the relabelled transitions in Eq. 2.7 to train the single-task BCQ policy in the first phase, which we do not since we focus on task inference in this work.

## 2.3 Experiment Results

We demonstrate the performance of our proposed algorithm (Sec. 2.3.1) and ablate the different design choices (Sec. 2.3.2). Sec. 2.3.3 shows that the multi-task policy can serve as a good initialization, significantly speeding up training on unseen tasks.

### 2.3.1 Performance evaluation on unseen tasks

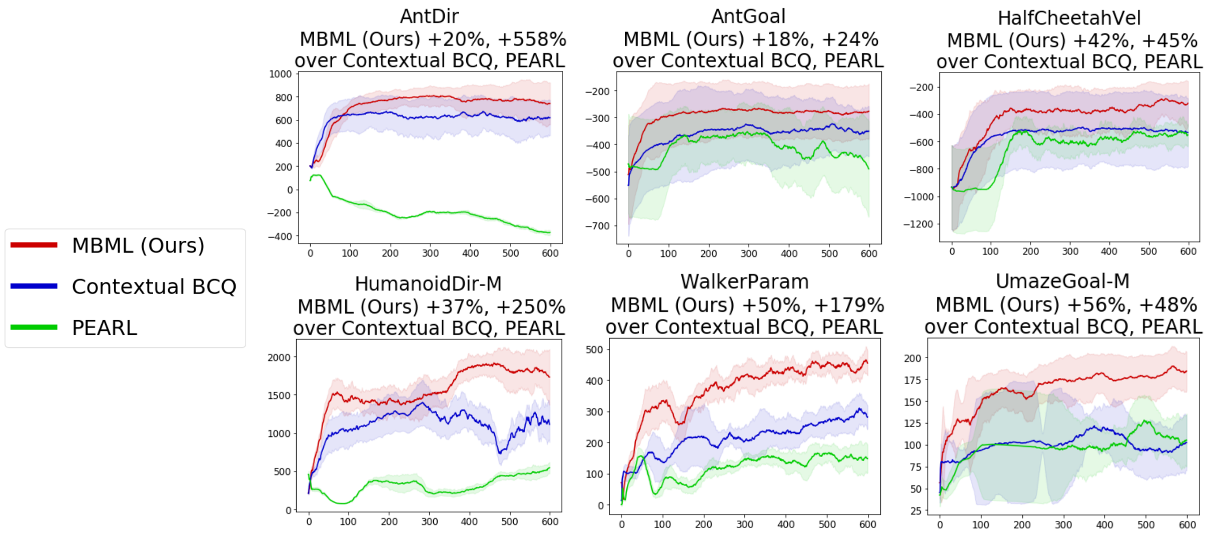


**Figure 2.3**: Results on unseen test tasks. x-axis is training epochs. y-axis is average episode returns. The shaded areas denote one std.

We evaluate in five challenging task distributions from MuJoCo [TET12] and a modified task distribution UmazeGoal-M from D4RL [FKN$^+$20b]. In AntDir and HumanoidDir-M, a target direction defines a task. The agent maximizes returns by running with maximal speed in the target direction. In AntGoal and UmazeGoal-M, a task is defined by a goal location, to which the agent should navigate. In HalfCheetahVel, a task is defined as a constant velocity the agent should achieve. We also consider the WalkerParam environment where random physical parameters parameterize the agent, inducing different transition functions in each task. The state for each task distribution is the OpenAI gym state. We do not include the task-specific information, such as the goal location or the target velocity in the state. The target directions and goals are sampled from a $120^\circ$ circular arc.

We argue that the version of HumanoidDir used in prior works does not represent a meaningful task distribution, where a single task policy can already achieve the optimal performance on unseen tasks. We thus modify the task distribution so that a policy has to infer the task identity to perform well, and denote it as HumanoidDir-M.

There are two natural baselines. The first is by modifying PEARL [RZQ+19] to train from the batch, instead of allowing PEARL to collect more transitions. We thus do not execute line $1 - 10$ in Algorithm 1 in the PEARL paper. On line 13, we sample the context and the RL batch uniformly from the batch. The second baseline is Contextual BCQ. We modify the networks in BCQ to accept the inferred task identity as input. We train the task inference module using the gradient of the value function loss. MBML and the baselines have the same network architecture. We are very much inspired by PEARL and BCQ. However, we do not expect PEARL to perform well in our setting because it does not explicitly handle the difficulties of learning from a batch without interactions. We also expect that our proposed algorithm will outperform Contextual BCQ thanks to more robust task inference.

We measure performance by the average returns over unseen tasks, sampled from the same task distribution. We do not count the first two episodes' returns [RZQ+19]. We obtain the batch for each training task by training Soft Actor Critic (SAC) [HZAL18a] with a fixed number of environment interactions.

From Fig. 2.3, MBML outperforms the baselines by a healthy margin in all task distributions. Even though PEARL does not explicitly handle the challenge of training from an offline

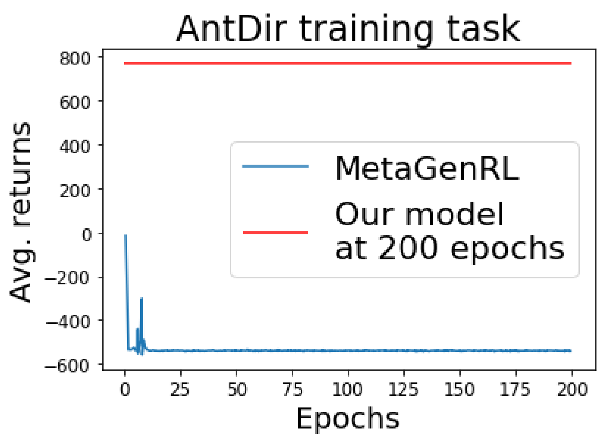


**Figure 2.4**: MetaGenRL quickly diverges and does not recover. Results obtained from official MetaGenRL code.

batch, it is remarkably stable, only diverging in AntDir. Contextual BCQ is stable, but converges to a lower performance than MBML in all task distributions. An astude reader will notice the issue of overfitting, for example Contextual BCQ in HumanoidDir-M. Since our paper is not about determining early stopping conditions and to ensure fair comparisons among the different algorithms, we compute the performance comparisons using the best results achieved by each algorithm during training.

We also compare with MetaGenRL [KvSS19]. Since it relies on DDPG [LHP$^+$15] to estimate value functions, which diverges in Batch RL [FMP19], we do not expect it to perform well in our setting. Fig. 2.4 confirms this, where its performance quickly plummets and does not recover with more training. Combining MetaGenRL and MBML is interesting since MetaGenRL generalizes to out-of-distribution tasks.

### 2.3.2 Ablations

We emphasize that our contributions lie in the triplet loss design coupled with transitions relabelling. Below, we provide ablation studies to demonstrate that both are crucial to obtain superior performance.

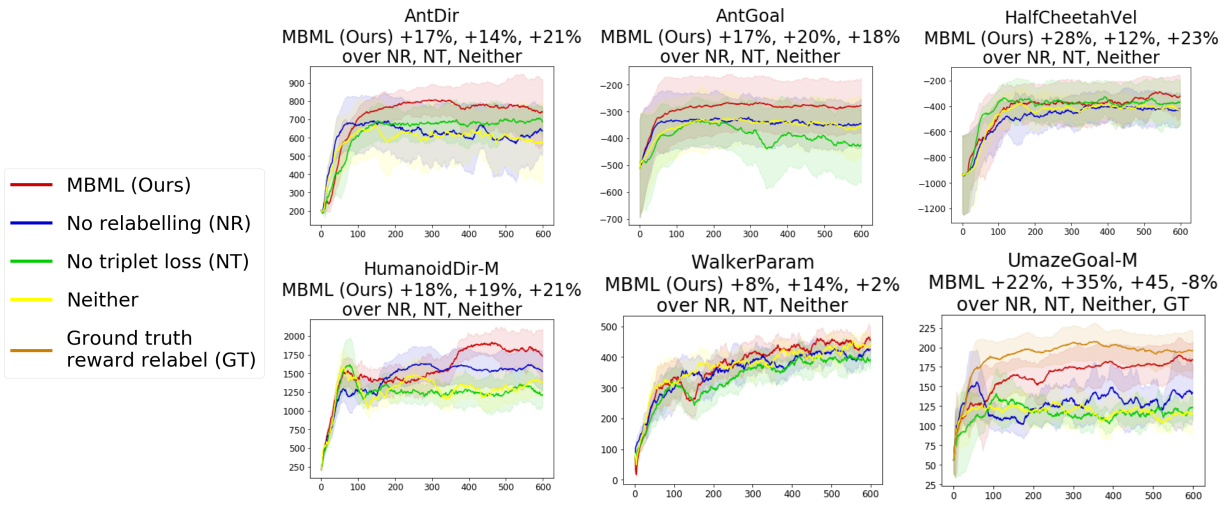


**Figure 2.5**: Ablation study. x-axis is training epochs. y-axis is average episode returns. The shaded areas denote one std.

**No relabelling.** To obtain hard negative examples, we search over a mini-batch to find the hardest positive-anchor and negative-anchor pairs, a successful and strong baseline from metric learning [HBL17]. This requires sampling $N$ context sets $\{\mathbf{c}_i^n\}_{n=1}^N$ for each task $i$, where $n$ indexes the context sets sampled for each task. Let $K$ be the number of training tasks, the triplet loss is:

$$\frac{1}{K}\sum_{i=1}^{K}\left[\max_{n,n'=1,\dots,N} d\left(q_\phi(\mathbf{c}_i^n), q_\phi(\mathbf{c}_i^{n'})\right) - \min_{\substack{n,n'=1,\dots,N\\ j=1,\dots,K, j\neq i}} d\left(q_\phi(\mathbf{c}_i^n), q_\phi(\mathbf{c}_j^{n'})\right) + m\right]_+ . \tag{2.11}$$

The *max* term finds the positive-anchor pair for task $i$ by considering every pair of context sets from task $i$ and selecting the pair with the largest divergence in the posterior over task identities. The *min* term finds the negative-anchor pair for task $i$ by considering every possible pair between the context sets sampled for task $i$ and the context sets sampled for the other tasks. It then selects the pair with the lowest divergence in the posterior over task identities as the negative-anchor pair.

**No triplet loss.** We train the task inference module using only gradient of the value function distillation loss (Eq. 2.3). To use the relabelled transitions, the module also takes as input the relabelled transitions during training. More concretely, given the context set $\mathbf{c}_i$ from task $i$, we sample an equal number of relabelled transitions from the other tasks $\tilde{\mathbf{c}}_i \sim \cup_j \mathbf{c}_{j\to i}$. During

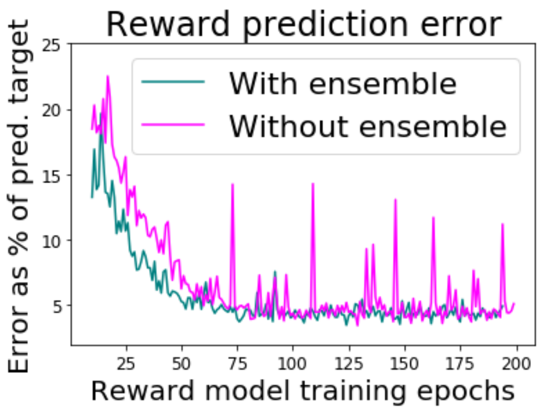


**Figure 2.6**: Reward prediction error on unseen task.

training, the input to the task inference module is the union of the context set $\mathbf{c}_i$ and the sampled relabelled transitions $\tilde{\mathbf{c}}_i$. In the full model, we also perform similar modification to the input of the module during training.

**No transition relabelling and no triplet loss.** This method is a simple combination of a task inference module and the distillation process. We refer to this algorithm as **Neither** in the graphs.

Fig. 2.5 compares our full model and the ablated versions. Our full model obtains higher returns than most of the ablated versions. For WalkerParam, our full model does not exhibit improvement over **Neither**. However, from Fig. 2.3, our full model significantly outperforms the baselines. We thus conclude that, in WalkerParam, the improvement over the baselines comes from distillation.

Comparing to the **No triplet loss** ablation, transition relabelling leads to more efficient computation of the triplet loss. Without the relabelled transitions, computing Eq. 2.11 requires $O(K^2N^2)$. Our loss in Eq. 2.9 only requires $O(K^2)$. We also need to relabel the transitions only once before training the multi-task policy. It is also trivial to parallelize across tasks.

We also study reward estimation accuracy. Fig. 2.6 shows that our reward model achieves low error on state-action pairs from another task, both with and without an ensemble. We also compare MBML against an ablated version that uses the ground truth reward function for relabelling on UmazeGoal-M. The model trained using the ground truth reward function only performs slightly better than the model trained using the learned reward function.

### 2.3.3 Using the multi-task policy to enable faster convergence when training on unseen tasks

While the multi-task policy generalize to unseen tasks, its performance is not optimal. If we allow further training, initializing networks with our multi-task policy significantly speeds up convergence to the optimal performance.

The initialization process is as followed. Given a new task, we use the multi-task policy to collect 10K transitions. We then train a new policy to imitate the actions taken by maximizing their log likelihood. As commonly done, the new policy outputs the mean and variance of a diagonal Gaussian distribution. The new policy does not take a task identity as input. The task inference module infers a task identity $\mathbf{z}$ from the 10K transitions. Fixing $\mathbf{z}$ as input, the distilled value function $Q_D$ initializes the new value function. Given the new policy and the initialized value function, we train them with SAC by collecting more data. To stabilize training, we perform target policy smoothing [FvHM18] and double-Q learning [Has10] by training two identically initialized value functions with different mini-batches.

Fig. 2.7 compares the performance of the policies initialized with our multi-task policy to randomly initialized policies. Initializing the policies with the MBML policy significantly increases convergence speed in all five task distributions, demonstrating our method's robustness. Even in the complex HumanoidDir-M task distribution, our method significantly speeds up the convergence, requiring only 85K environment interactions, while the randomly initialized policies require 350K, representing a 76% improvement in sample efficiency. Similar conclusions hold when comparing against randomly initialized SAC where the two value functions are trained using different mini-batches. We also note that our initialization method does not require extensive hyper-parameter tuning.

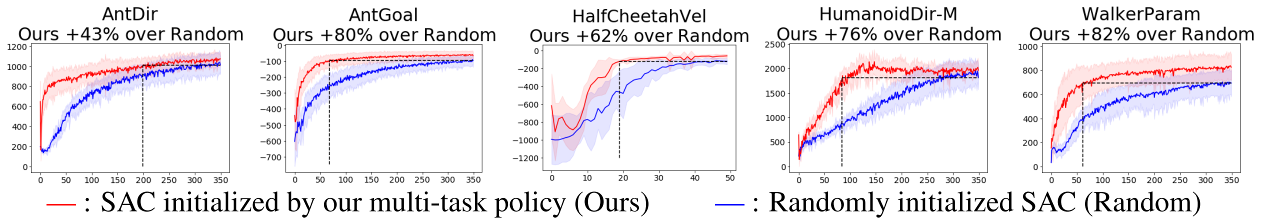


**Figure 2.7**: Initialization results. x-axis is number of interactions in thousands. y-axis is the average episode returns over unseen tasks. The shaded areas denote one std.

## 2.4 Related Works

**Batch RL** Recent advances in Batch RL [ASN19, KFS+19a, FMP19, CZW+19, KZTL20] focus on the single-task setting, which does not require training a task inference module. Thus they are not directly applicable to the Multi-task Batch RL. [SSB+20, CCN+20] also consider the multi-task setting but assume access to the ground truth task identity and reward function of the test tasks. Our problem setting also differs, where the different training batches do not have significant overlap in state-action visitation frequencies, leading to the challenge of learning a robust task inference module.

**Task inference in multi-task setting** The challenge of task inference in a multi-task setting has been tackled under various umbrellas. Meta RL [RZQ+19, ZSI+20, FCSS19, HGH+19, LLGW19, SHD18, ZSK+19] trains a task inference module to infer the task identity from a context set. We also follow this paradigm. However, our setting presents additional challenge to train a robust task inference module, which motivates our novel triplet loss design. As the choice of loss function is crucial to train an successful task inference module in our settings, we will explore the other loss functions, e.g. loss functions discussed in [RMS+20], in future work. Other multi-task RL works [ESM+18a, YXWW20, YKG+19, DTB+19] focus on training a good multi-task policy, rather than the task inference module, which is an orthogonal research direction to ours.

**Meta RL** Meta RL [LLGW19, WKT+16, DSC+16, FAL17, NAS18, HCI+18] optimizes for quick adaptation. However, they require interactions with the environment during training.

Even though we do not explicitly optimize for quick adaptation, we demonstrate that initializing a model-free RL algorithm with our policy significantly speeds up convergence on unseen tasks. [FCSS19] uses the data from the training tasks to speed up convergence when learning on new tasks by propensity estimation techniques. This approach is orthogonal to ours and can potentially be combined to yield even greater performance improvement.

## 2.5 Discussions

The issue of learning the wrong dependency does not surface when multi-task policies are tested in Atari tasks because their state space do not overlap [PBS15, HSE$^+$19, ESM$^+$18b]. Each Atari task has distinctive image-based state. The policy can perform well even when it only learns to correlate the state to the task identity. When Mujoco tasks are used to test online multi-task algorithms [ZSI$^+$20, FCSS19], the wrong dependency becomes self-correcting. If the policy infers the wrong task identity, it will collect training data which increases the overlap between the datasets of the different training tasks, correcting the issue overtime. However, in the batch setting, the policy can not collect more transitions to self-correct inaccurate task inference. Our insight also leads to exciting possibility to incorporate mechanism to quickly infer the correct causal relationship and improve sample efficiency in Multi-task RL, similar to how causal inference method has motivated new innovations in imitation learning [dHJL19].

Our first limitation is the reliance on the generalizability of simple feedforward NN. Future research can explore more sophisticated architecture, such as Graph NN with reasoning inductive bias [XLZ$^+$19, SGT$^+$08, WPC$^+$20, ZCZ$^+$18] or structural causal model [Pea10, P$^+$09], to ensure accurate task inference. We also assume the learnt reward function of one task can generalize to state-action pairs from the other tasks, even when their state-action visitation frequencies do not overlap significantly. To increase the prediction accuracy, we use a reward ensemble to estimate epistemic uncertainty. We note that the learnt reward functions do not need to generalize to every state-action pairs, but only enough pairs so that the task inference module is forced to consider the rewards when trained to minimize Eq. 2.8. Crucially, we do not need to solve the task inference challenge while learning the reward functions and using them for relabelling, allowing us to side-step the challenge of task inference.

The second limitation is in scope. We only demonstrate our results on tasks using proprioceptive states. Even though they represent high-dimensional variables in a highly nonlinear ODE, the model does not need to tackle visual complexity. The tasks we consider also have relatively dense reward functions and not binary reward functions. These tasks, such as navigation and running, are also quite simple in the spectrum of possible tasks we want an embodied agents to perform. These limitations represent exciting directions for future work.

Another interesting future direction is to apply supervised learning self-distillation techniques [XLHL19, MFB20], proven to improve generalization, to further improve the distillation procedure. To address the multi-task learning problem for long-horizon tasks, it would also be beneficial to consider skill discovery and composition from the batch data [PCZ$^+$19, SAL$^+$20]. However, in this setting, we still need effective methods to infer the correct task identity to perform well in unseen tasks. Our explanation in Sec. 2.2 only applies when the tasks differ in reward function. Extending our approach to task distributions with varying transition functions is trivial. Sec. 2.3 provide experimental results for both cases.

## 2.6 Acknowledgement

We thank Fangchen Liu and Chaochao Lu for discussions and proofreading of the paper. Computing needs were supported by the Nautilus Pacific Research Platform. Chapter 2, in part, is a reprint of material from Jiachen Li, Quan Vuong, Shuang Liu, Minghua Liu, Kamil Ciosek, Keith Ross, Henrik Iskov Christensen, Hao Su. “Multi-task Batch Reinforcement Learning with Metric Learning”. In Advances in Neural Information Processing Systems, 2020. The dissertation

author supervised Jiachen Li, which was a Master student at UCSD at the time of pulication, and is also the primary author of the paper.

# Chapter 3

# Dual Generator Offline Reinforcement Learning

Offline reinforcement learning (RL) algorithms aim to extract policies from datasets of previously logged experience. The promise of offline RL is to extract *decision making engines* from existing data [LKTF20]. Such promise is especially appealing in domains where data collection is expensive or dangerous, but large amounts of data may already exists (e.g., robotics, autonomous driving, task-oriented dialog systems). Real-world datasets often consist of both expert and sub-optimal behaviors for the task of interest and also include potentially unrelated behavior corresponding to other tasks. While not all behaviors in the dataset are relevant for solving the task of interest, even sub-optimal trajectories can provide an RL algorithm with some useful information. In principle, if offline RL algorithms can combine segments of useful behavior spread across multiple sub-optimal trajectories together, the combined segments can then perform better than any behavior observed in the dataset.

Effective offline RL requires estimating the value of actions other than those that were taken in the dataset, so as to pick actions that are better than the actions selected by the behavior policy. However, this requirement introduces a fundamental tension: the offline RL method must generalize to new actions, but it should not attempt to use actions in the Bellman backup for which the value simply cannot be estimated using the provided data. These are often referred to in the literature as out-of-distribution (OOD) actions [KFS$^+$19b]. While a wide variety of methods have been proposed to constrain offline RL to avoid OOD actions [KTFN21, FMP19, ASN19], the formulation and enforcement of such constraints can be challenging, and might introduce considerable complexity, such as the need to explicitly estimate the behavior policy [WTN19] or evaluate high-dimensional integrals [KZTL20]. Generative adversarial networks (GANs) in principle offer an appealing and simple solution: use the discriminator to estimate whether an action is in-distribution, and train the policy as the "generator" in the GAN formulation to fool this discriminator. Although some prior works have proposed variants on this approach [WTN19], it has been proven difficult in practice as GANs can already suffer from instability when the discriminator is too powerful. Forcing the generator (i.e., the policy) to simultaneously *both* maximize reward and fool the discriminator only exacerbates the issue of an overpowered discriminator.

We propose a novel solution that enables the effective use of GANs in offline RL, in the process not only mitigating the above challenge but also providing a more appealing form of support constraint that leads to improved performance. Our key observation is that the generative distribution in GANs can be split into *two* separate distributions, one that represents the "good parts" of the data distribution and becomes the final learned policy, and an auxiliary generator that becomes the policy's complement, such that the mixture of the two is equal to the data distribution. This formulation removes the tension between maximizing rewards and matching the data distribution perfectly: as long as the learned policy is within the *support* of the data distribution, the complement will pick up the slack and model the "remainder" of the data distribution, allowing the two generators together to perfectly fool the discriminator. If however the policy ventures outside of the support of the data, the second generator cannot compensate for this mistake, and the discriminator will push the policy back inside the support. We name our method DASCO, for **D**ual-Generator **A**dversarial **S**upport **C**onstrained **O**ffline RL.

Experimentally, we demonstrate the benefits of our approach, DASCO, on standard benchmark tasks. For offline datasets that consist of a combination of expert, sub-optimal and noisy data, our method outperforms distribution-constrained offline RL methods by a large margin.

## 3.1 Related Work

Combining behaviors from sub-optimal trajectories to obtain high-performing policies is a central promise of offline RL. During offline training, querying the value function on unseen actions often leads to value over-estimation and unrecoverable collapse in learning progress. To avoid querying the value functions on out-of-distribution actions, existing methods encourage the learned policies to match the distribution of the dataset generation policies. This principle has been realized with a variety of practical algorithms [JGS+19, WTN19, PKZL19, SSB+20, WTN19, KFS+19a, KTFN21, KNL21, WNŻ+20, FG21, CZW+19, FMG22, JLK22, MWY+22, DBSV22, LTLL22].

For example, by optimizing the policies with respect to a conservative lower bound of the value function estimate [KZTL20], only optimizing the policies on actions contained in the dataset [KTFN21], or jointly optimizing the policy on the long-term return and a behavior cloning objective [FG21]. While *explicitly* enforcing distribution constraint by adding the behavior cloning objective allows for good performance on near-optimal data, this approach fails to produce good trajectories on sub-optimal datasets [KTFN21]. Methods that *implicitly* enforce distribution constraints, such as CQL and IQL, have seen more successes on such datasets. However, they still struggle to produce near-optimal trajectories when the actions of the dataset generation policies are corrupted with noise or systematic biases (a result we demonstrate in Section 3.4).

However, enforcing distribution constraints to avoid value over-estimation may not be necessary. It is sufficient to ensure the learned policies do not produce actions that are too unlikely under the dataset generation policy. That is, it is not necessary for the learned policy to fully *cover* the data distribution, only to remain in-support [KFS+19a, Kum19, LKTF20, WTN19, ZBH20, CGY+22]. Unfortunately, previous methods that attempt to instantiate this principle into algorithms have not seen as much empirical success as algorithms that penalize the policies for not matching the action distribution of the behavior policies. In this paper, we propose a new GAN-based offline RL algorithm whose use of dual generators naturally induce support constraint and has competitive performance with recent offline RL methods. In a number of prior works, GANs have been used in the context of imitation learning to learn from expert data [HE16, LSE17, HCS+17, LZF+21]. In this work, we show that dual-generator GANs can be used to learn from sub-optimal data in the context of offline RL.

## 3.2 Background

Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ define a Markov decision process (MDP), where $\mathcal{S}$ and $\mathcal{A}$ are state and action spaces, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ is a state-transition probability function, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function and $\gamma$ is a discount factor. Reinforcement learning methods aim at finding a policy $\pi(a|s)$ that maximizes the expected discounted reward $R(\tau) = \sum_{t=0}^{T} \gamma^t R(s_t, a_t)$ over trajectories $\tau = (s_0, a_0, \ldots, s_T, a_T)$ with time horizon $T$ induced by the policy $\pi$.

In this work, we concentrate on the offline or off-policy RL setting, i.e. finding an optimal policy given a dataset $\mathcal{D}$ of previously collected experience $\tau \sim \mathcal{D}$ by a behavior policy $\pi_\beta$. A particularly popular family of methods for offline learning are based on training a Q-function through dynamic programming using temporal-difference (TD) learning [WD92, SB18]. Such methods train a Q-function to satisfy the Bellman equation:

$$Q(s_t, a_t) = R(s_t, a_t) + \gamma \mathbb{E}_{a \sim \pi}[Q(s_{t+1}, a)].$$

$\pi(a|s) = \arg\max_a Q_\theta(s,a)$ In Q-learning, the policy is replaced with a maximization, such that $\pi(a|s) = \arg\max_a Q_\theta(s,a)$, while actor-critic methods optimize a separate parametric policy $\pi_\phi(a|s)$ that maximizes the Q-function. In this work, we extend the Soft Actor-Critic (SAC) method [HZAL18a] for learning from diverse offline datasets.

Generative Adversarial Networks (GANs) [GPAM⁺14] enable modeling a data distribution $p_{\mathcal{D}}$ through an adversarial game between a generator $G$ and a discriminator $D$:

$$\min_G \max_D \mathbb{E}_{x\sim p_{\mathcal{D}}}[\log(D(x))] + \mathbb{E}_{z\sim p(z)}[\log(1-D(G(z)))] \tag{3.1}$$

For this two player zero-sum game, [GPAM⁺14] shows that for a fixed generator $G$, the optimal discriminator is $D^*_G(x) = \dfrac{p_{\mathcal{D}}(x)}{p_{\mathcal{D}}(x) + p_G(x)}$ and the optimal generator matches the data distribution $p^*_g(x) = p_{\mathcal{D}}$.

GAN has been extended to the offline RL setting by interpreting the discriminator function as a measure of how likely an action is under the behavior policy, and jointly optimizing the policy to maximize an estimate of the long-term return and the discriminator function [WTN19]:

$$\min_\pi \max_D \mathbb{E}_{s,a\sim p_{\mathcal{D}}}[\log(D(s,a))] + \mathbb{E}_{s\sim p_{\mathcal{D}},a\sim\pi(a|s)}[\log(1-D(s,a))] - \mathbb{E}_{s\sim p_{\mathcal{D}},a\sim\pi(a|s)}[Q(s,a)], \tag{3.2}$$

where $Q(s,a)$ is trained via the Bellman operator to approximate the value function of the policy $\pi(a|s)$. This leads to iterative policy evaluation and policy improvement rules for the actor and the policy [WTN19]. During the $k^{th}$ update step, given the most recent values for the policy $\pi^k$, the value function $Q^k$, and the discriminator $D^k$, we perform the following updates to obtain the next values for the value function and the policy:

$$
\begin{aligned}
Q^{k+1} &\leftarrow \arg\min_{Q} \mathop{\mathbb{E}}_{s,a,s'\sim\mathcal{D}} \left[ \left( (R(s,a) + \gamma \mathop{\mathbb{E}}_{a'\sim\pi^k(a'|s')} [Q^k(s',a')]) - Q_{target}(s,a) \right)^2 \right] \\
\pi^{k+1} &\leftarrow \arg\max_{\pi} \mathop{\mathbb{E}}_{s\sim\mathcal{D},a\sim\pi^k(a|s)} \left[ Q^{k+1}(s,a) + \log D^k(s,a) \right]
\end{aligned}
\tag{3.3}
$$

where the $\log D(a|s)$ term in the policy objective aims at regularizing the learnt policy to prevent it from outputting OOD actions. In practice, training the policy to maximize both the value function and discriminator might lead to conflicting objectives for the policy and thus poor performance on either objective. This can happen when the data contains a mixture of good and bad actions. Maximizing the value function would mean avoiding low-reward behaviors. On the other hand, maximizing the discriminator would require outputting all in-distribution actions, including sub-optimal ones. Our approach alleviates this conflict and enables *in support* maximization of the value function when learning from mixed-quality datasets.

## 3.3 Dual-Generator Adversarial Support Constraint Offline RL

We now present our algorithm, which uses a novel dual-generator GAN in combination with a weighting method to enable GAN-based offline RL that constrains the learned policy to remain within the support of the data distribution. We call our method *Dual-generator Adversarial Support Constraint Offline RL (DASCO)*. We will first introduce the dual-generator training method generically, for arbitrary generators that must optimize a user-specified function $f(x)$ within the support of the data distribution in Section 3.3.1. We will then show this method can be incorporated into a complete offline RL algorithm in Section 3.3.2 in combination with our proposed weighting scheme, and then summarize the full resulting actor-critic method in Section 3.3.3.

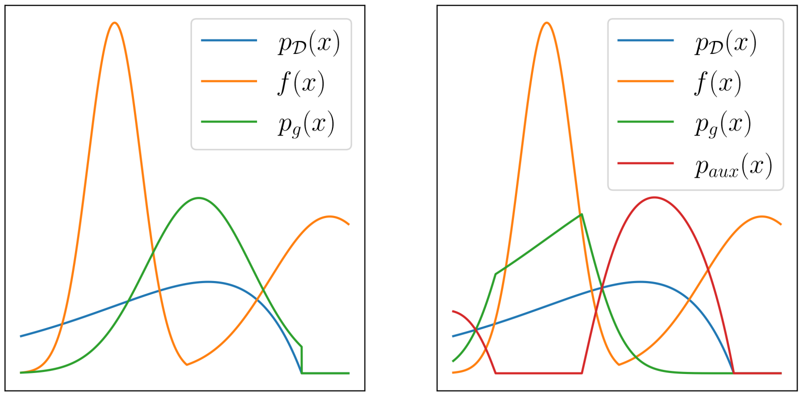


**Figure 3.1**: Visualizations to illustrate the benefit of *dual* generators over single generator when maximizing a secondary objective $f(x)$ in the GAN framework. In both figures, $p_{\mathcal{D}}(x)$ is the data distribution. The x-axis is a one-dimensional sample space. **Left:** In this figure, since there is only a single generator, the generator $G$ is trained to jointly maximize the objective $f(x)$ and matches the data distribution $p_{\mathcal{D}}(x)$. The distribution $p_G$ induced by the generator is thus not very good at either maximizing the objective $f(x)$ or matching the data distribution. **Right:** In this figure, we have two generators, inducing two distributions $p_G$ and $p_{aux}$. By introducing the auxiliary generator $G_{aux}$ into the GAN framework, the primary generator can better maximize the objective $f(x)$ while staying within the support of the data distribution $p_{\mathcal{D}}$. The mixed distribution also perfectly matches the data distribution, i.e. $\frac{p_g(x)+p_{aux}(x)}{2} = p_{\mathcal{D}}(x)$. Note that in these two figures, the primary generator aims to maximize $f(x)$ (instead of minimize) to allow for more intuitive interpretation.

### 3.3.1 Dual generator in-support optimization

In this section, we will develop an approach for performing a joint optimization of adversarial and secondary objectives of the generator in a GAN framework, which we will then apply to offline RL. This is a necessary component for performing the joint optimization in Eq. 3.2 without introducing a conflict of these objectives. All proofs for theorems presented in this section are in Appendix A.

Let's consider a general objective that requires training a generator $G$ to fool the discriminator $D$ while also optimizing the expected value of some other function $f$:

$$\begin{aligned}\min_G \max_D \quad & \mathbb{E}_{x\sim p_{\mathcal{D}}}[\log(D(x))] \\ & + \mathbb{E}_{z\sim p(z)}[\log(1-D(G(z)))] \\ & + \mathbb{E}_{z\sim p(z)}[f(G(z))] \end{aligned} \tag{3.4}$$

where the first two terms are the same as the objective of the GAN formulation. We have also added an additional term $\mathbb{E}_{z\sim p(z)}[f(G(z))]$, where $f$ is a mapping from the generator output to a scalar value. The third term represents a secondary objective that the generator should optimize.

**Theorem 3.3.1.** *The optimal generator of Eq. 3.4 induces a distribution* $p_g^*(x) = p_{\mathcal{D}}(x)\frac{e^{-f(x)-\nu}}{2-e^{-f(x)-\nu}}$, *where* $\nu > 0$ *is the Lagrange multiplier that ensures that* $p_g^*(x)$ *is normalized to 1.*

We can see that by adding a secondary objective function for the generator, in general, the optimal generator does not attempt to match the data distribution $p_{\mathcal{D}}(x)$ anymore, but instead tries to match the data distribution weighted by $\frac{e^{-f(x)-\nu}}{2-e^{-f(x)-\nu}}$. We expect that in such case, the discriminator clearly has an advantage in the two player zero-sum game and will be able to distinguish between real samples and sample generated by the generator.

To allow the generator to specialize in optimizing the secondary objective function, we propose to introduce a second auxiliary generator that matches the portion of the data distribution that is not well captured by the primary generator. Let $p_{mix} = \frac{p_g + p_{aux}}{2}$, consider the min-max problem:

$$\min_{G,G_{aux}} \max_D \mathbb{E}_{x\sim p_{\mathcal{D}}}[\log(D(x))] + \mathbb{E}_{x\sim p_{mix}}[\log(1-D(x))] + \mathbb{E}_{x\sim p_g}[f(x)], \tag{3.5}$$

where we mix samples from the primary generator $G$ and the auxiliary generator $G_{aux}$ to generate samples that can fool the discriminator. The mixing is indicated by the distribution $p_{mix}$ in the second term of Eq. 3.5. The first and third term of Eq. 3.5 are the same as the objective in Eq. 3.4.

We next theoretically demonstrate the benefit of adding the auxiliary generator to the GAN formulation with the following Theorem.

**Theorem 3.3.2** (Informal). *The primary generator $p_G$ performs in-support optimization of $f(x)$.*

We first note that the optimal solution of the mixed distribution from Eq. 3.5 is the real data distribution:

$$\frac{p^*_{aux}(x) + p^*_g(x)}{2} = p_{\mathcal{D}}(x) \tag{3.6}$$

Accordingly, the optimal auxiliary generator distribution can be expressed as

$$p^*_{aux}(x) = 2p_{\mathcal{D}}(x) - p^*_g(x) \tag{3.7}$$

Let $x_0$ to be the element inside the support of the data distribution $p_{\mathcal{D}}$ that minimizes $f$. That is:

$$x_0 = \underset{x \in \mathrm{Supp}(p_{\mathcal{D}})}{\arg\min} \; f(x)$$

When optimizing the secondary objective $f(x)$, the primary generator will maximize the probability mass of in-support samples that maximize $f(x)$. However, Eq. 3.7 introduces a constraint that enforces $2p_{\mathcal{D}}(x) - p^*_g(x) \geq 0$ for $p^*_{aux}(x) \geq 0$ to remain a valid distribution. This leads us to conclude that the optimal primary generator $p^*_g$ assigns the following probability to $x_0$:

$$p^*_g(x_0) = \begin{cases} 2p_{\mathcal{D}}(x_0) & \text{if} \quad 2p_{\mathcal{D}}(x_0) < 1 \\ 1 & \text{otherwise} \end{cases} \tag{3.8}$$

Interestingly, if the global maximum $x_0$ is not taking the full probability mass, the rest of the probability mass is redistributed to the next best in-support maxima, which we can define recursively:

$$\text{For } x_i \in \underset{x \in \text{Supp}(p_{\mathcal{D}}) \setminus \{x_j\}_{j=0}^{i-1}}{\arg\min} f(x),\ p_g^*(x_i) = \begin{cases} 2p_{\mathcal{D}}(x_i) & \text{if} \quad \sum_{j=0}^{i} p_g^*(x_j) < 1 \\ 1 - \sum_{j=0}^{i-1} p_g^*(x_j) & \text{if} \quad \sum_{j=0}^{i} p_g^*(x_j) > 1 \\ 0 & \text{if} \quad \sum_{j=0}^{i-1} p_g^*(x_j) = 1 \end{cases} \tag{3.9}$$

We provide more explanation for the solution in Eq. 3.9. In the first case, $p_g^*(x_i) = 2p_{\mathcal{D}}(x_i)$ if $\sum_{j=0}^{i} p_g^*(x_j) < 1$. That is, if the optimal solution for the primary generator $p_g^*$ *can* assign the probability $2p_{\mathcal{D}}(x_i)$ to the $i^{th}$ in support minima of $f(x)$ without the total sum of probability assigned $\sum_{j=0}^{i} p_g^*(x_j)$ going over 1, then the primary generator $p_g^*$ will assign the probability $2p_{\mathcal{D}}(x_i)$ to $x_i$.

In the second case, $p_g^*(x_i) = 1 - \sum_{j=0}^{i-1} p_g^*(x_j)$ if $\sum_{j=0}^{i} p_g^*(x_j) > 1$. That is, if by assigning the probability $2p_{\mathcal{D}}(x_i)$ to the $i^{th}$ in support minima of $f(x)$, the total sum of probability assigned $\sum_{j=0}^{i} p_g^*(x_j)$ *goes over* 1, then the primary generator $p_g^*$ will assign the remaining probability $1 - \sum_{j=0}^{i-1} p_g^*(x_j)$ to $x_i$. In the third case, the generator assigns probability 0 to $x_i$ because all the probability has already been assigned.

To summarize the benefit of dual generator, we note that by introducing an auxiliary generator and mixing it with the primary generator, not only does the optimal solution for the mixed distribution match the real data distribution, but also the primary generator can better optimize the secondary objective $f$ on the part of the domain of $f$ that is within the support of the data distribution $p_{\mathcal{D}}$. To better illustrate the benefit, we provide a visual explanation of the benefit in Figure 3.1.

## 3.3.2 Update rules for offline reinforcement learning

We will now incorporate the dual-generator method to train policies for offline RL, based on optimizing the joint objective from Eq. 3.5. The updates for the actor and the critic are generally similar to Eq. 3.3. However, simply combining Eq. 3.5 and Eq. 3.3 can still allow the policy to exploit errors in the value function during the policy improvement step. We therefore augment the policy improvement step with an adaptive weight on the Q-value. More concretely, as the policy improvement step samples actions from the current policy iterate $\pi^k$ to optimize the policy objective, there is a non-zero probability that the sampled actions will exploit spurious maxima in the value function and have their probability of being sampled again in the future increased. If the same actions are sampled during the policy evaluation step, the errors in the value functions from the next states are backed up into the preceding states, leading to divergent value functions, as we observe in our experiments. To alleviate this issue, we use the probability assigned to the sampled actions by the discriminator to weight the value function estimates in the policy objective, leading to the following updates:

$$Q^{k+1} \leftarrow \arg\min_{Q} \mathop{\mathbb{E}}_{s,a,s' \sim \mathcal{D}} \left[ \left( (R(s,a) + \gamma \mathop{\mathbb{E}}_{a' \sim \pi^k(a'|s')} [Q^k(s',a')]) - Q_{target}(s,a) \right)^2 \right] \tag{3.10}$$

$$\pi^{k+1} \leftarrow \arg\max_{\pi} \mathop{\mathbb{E}}_{s,a_{\mathcal{D}} \sim \mathcal{D}, a \sim \pi^k(a|s)} \left[ \frac{D^k(s,a)}{D^k(s,a_{\mathcal{D}}(s))} Q^{k+1}(s,a) + \log D^k(s,a) \right], \tag{3.11}$$

where $a_{\mathcal{D}}(s)$ is the action from the offline dataset. The term $D^k(s,a)$ down-weights the contribution of the gradient of the value function to the policy update if the discriminator deems the sampled action too unlikely. We further calibrate the probability $D^k(s,a)$ by dividing it with the probability $D^k(s,a_{\mathcal{D}}(s))$ that the discriminator assigns to the dataset action $a_{\mathcal{D}}(s)$. It should be noted that during optimization the gradients are not propagated into these weights.

Next, we define the update rules for the auxiliary generator and the discriminator. We mix the samples from the $k^{th}$ iterate of the policy $\pi^k$ and the distribution $p_{aux}$ induced by the $k^{th}$ iterate of the auxiliary generator $G^k_{aux}$, that is, let $p_{mix} = \dfrac{\pi^k + p_{aux}}{2}$. At every iteration $k$, we update the

$k^{th}$ iterate of the auxiliary generator $G_{aux}^{k}$ and discriminator $D^{k}$ using the objectives:

$$G_{aux}^{k+1} \leftarrow \arg\min_{G_{aux}} \mathbb{E}_{x \sim p_{mix}}[\log\Big(1 - D^{k}(s,a)\Big)] \tag{3.12}$$

$$D^{k+1} \leftarrow \arg\max_{D} \mathbb{E}_{x \sim p_{\mathcal{D}}}[\log\Big(D^{k}(s,a)\Big)] + \mathbb{E}_{x \sim p_{mix}}[\log\Big(1 - D^{k}(s,a)\Big)] \tag{3.13}$$

### 3.3.3 Algorithm summary

Algorithm 2 provides a step-by-step description of our algorithm. At every training step, we sample a batch of transitions from the offline dataset and proceed to update the parameters of the value function, the policy, the auxiliary generator and the discriminator in that order.

**Algorithm 2** DASCO algorithm summary

```
1: Initialize Q-function $Q_\theta$, policy $\pi_\phi$, auxiliary generator $G_{aux,\psi}$, discriminator $D_\omega$
2: for training step $k$ in $\{1,\ldots,N\}$ do
3:     $(s,a,r,s') \leftarrow \mathcal{D}$: Sample a batch of transitions from the dataset
4:     $\theta^{k+1} \leftarrow$ Update Q-function $Q_\theta$ using the Bellman update in Eq. 3.10
5:     $\phi^{k+1} \leftarrow$ Update policy $\pi_\phi$ using the augmented objective in Eq. 3.11
6:     $\psi^{k+1} \leftarrow$ Update auxiliary generator $G_{aux,\psi}$ using the objective in Eq. 3.12
7:     $\omega^{k+1} \leftarrow$ Update discriminator $D_\omega$ using mixed samples from $\pi_\phi$ and $G_{aux,\psi}$ as in Eq. 3.13
8: end for
```

## 3.4 Experiments

Our experiments aim at answering the following questions:

1. When learning from offline datasets that require combining actions from sub-optimal trajectories, does DASCO outperform existing methods that are based on distribution constraints?

2. On standard benchmarks such as D4RL [FKN$^{+}$20a], how does DASCO compare against recent methods?

3. Are both the dual generator and the probability ratio weight important for the performance of DASCO?

### 3.4.1 Comparisons on standard benchmarks and new datasets

For our first set of experiments, we introduce four new datasets to simulate the challenges one might encounter when using offline RL algorithms on real world data. These datasets introduce additional learning challenges and require the algorithm to combine actions in different trajectories to obtain good performance. We use the existing AntMaze environments from the D4RL suite [FKN$^+$20a]: antmaze-medium and antmaze-large. In these two environments, the algorithm controls an 8-DoF "Ant" quadruped robot to navigate a 2D maze to reach desired goal locations. The D4RL benchmark generates the offline datasets for these two environments using two policies: 1. a low-level goal reaching policy that outputs torque commands to move the Ant to a nearby goal location and 2. a high-level waypoint generator to provide sub-goals that guide the low-level goal-reaching policy to the desired location. We use the same two policies to generate two new classes of datasets.

For the `noisy` dataset, we add Gaussian noise to the action computed by the low-level goal-reaching policy. The noise variance depends on the 2D location of the Ant in the maze – larger in some 2D regions than others. We intend this dataset to be representative of situations where the data generation policies are more deterministic in some states than others [KHSL22] – a robot picking up an object has many good options to approach the object, but when the robot grasps the object, its behavior should be more deterministic to ensure successful grasp without damaging or dropping the object [MSZ94].

For a `biased` dataset, in addition to adding Gaussian noise to the actions as it is done in the `noisy` dataset, we also add bias to the actions computed by the low-level policy. The values of the bias also depend on the current 2D location of the Ant in the maze. This setting is meant to simulate learning from relabelled data, where the dataset was generated when the data

**Table 3.1**: Performance comparison to baselines when learning from the `noisy` and `biased` AntMaze datasets. Our method outperforms the baselines significantly. The value in parenthesis indicates the standard deviation of mean episode return, computed over 3 different runs.

| Dataset | BEAR | EDAC | CQL | IQL | DASCO (Ours) |
|---|---|---|---|---|---|
| antmaze-large-bias | - | - | 61.7 (3.5) | 41.0 (7.9) | 63.9 (6.0) |
| antmaze-large-noisy | - | - | 50.3 (2.3) | 39.0 (6.4) | 54.3 (2.0) |
| antmaze-medium-bias | 0.0 (0.0) | 0.0 (0.0) | 66.7 (2.9) | 48.0 (5.9) | 90.2 (2.4) |
| antmaze-medium-noisy | 0.0 (0.0) | 0.0 (0.0) | 55.7 (4.7) | 44.3 (1.7) | 86.3 (4.5) |
| `noisy` and `biased` antmaze-v2 total | - | - | 234.4 | 172.3 | **294.7** |

generation policies were performing a different task than the tasks we are using the dataset to learn to perform. Relabelling offline data is a popular method for improving the performance of offline RL algorithms [VLL$^+$19, SYF$^+$22], especially when we have much more data for some tasks than others [KVC$^+$21]. In the AntMaze environment, offline RL algorithms must combine data from sub-optimal trajectories to learn behaviors with high returns. In addition, `noisy` and `biased` datasets present a more challenging learning scenarios due to the added noise and systematic bias which vary non-uniformly based on the 2D location of the Ant.

Table 3.1 illustrates the performance comparison of our method and representative methods that enforce distribution constraints, either through optimizing a conservative lower bound of the value estimates (CQL) or only optimizing the policy on actions in the dataset using Advantage Weighted Regression [PKZL19] (IQL). Our method outperforms both CQL and IQL. In these tasks, to ensure a fair comparison between different methods, we perform oracle offline policy selection to obtain the performance estimates for CQL, IQL, and our method. We also compare the performance on standard AntMaze tasks when learning from the datasets in the D4RL benchmark without modifications in Table 3.2. In these tasks, our method outperforms IQL by a large margin on two diverse datasets.

By comparing the results in Table 3.1 (learning from `noisy` and `biased` datasets) and Table 3.2 (learning from existing offline datasets in D4RL), we also note that our proposed algorithm outperforms distribution-constraint offline RL algorithms (CQL, IQL) more consistently when tested on the `noisy` and `biased` datasets. For the results in these two tables, the definition

**Table 3.2**: Performance comparison to distribution-constrained baselines on AntMaze tasks in D4RL. Our algorithm outperforms the baselines when learning from the diverse and play datasets.

| Dataset | CQL | IQL | DASCO (Ours) |
|---|---|---|---|
| antmaze-umaze | 97.0 (0.8) | 90.3 (1.9) | 99.2 (0.0) |
| antmaze-umaze-diverse | 58.7 (12.2) | 70.3 (4.6) | 89.0 (1.7) |
| antmaze-medium-play | 77.0 (1.6) | 82.7 (0.5) | 92.3 (1.5) |
| antmaze-medium-diverse | 80.0 (0.0) | 82.3 (1.9) | 87.1 (0.4) |
| antmaze-large-play | 53.3 (4.6) | 55.7 (3.1) | 64.4 (1.7) |
| antmaze-large-diverse | 48.0 (2.9) | 50.0 (3.6) | 74.1 (2.8) |
| antmaze total | 414.0 | 431.3 | **506.1** |

**Table 3.3**: Performance comparison with recent offline RL algorithms on the Gym locomotion tasks

| Dataset | BC | 10%BC | DT | AWAC | Onestep RL | TD3+BC | CQL | IQL | DASCO (Ours) |
|---|---|---|---|---|---|---|---|---|---|
| halfcheetah-medium-replay | 36.6 | 40.6 | 36.6 | 40.5 | 38.1 | 44.6 | 45.5 | 44.2 | 44.7 |
| hopper-medium-replay | 18.1 | 75.9 | 82.7 | 37.2 | 97.5 | 60.9 | 95.0 | 94.7 | 101.7 |
| walker2d-medium-replay | 26.0 | 62.5 | 66.6 | 27.0 | 49.5 | 81.8 | 77.2 | 73.9 | 74.5 |
| halfcheetah-medium-expert | 55.2 | 92.9 | 86.8 | 42.8 | 93.4 | 90.7 | 91.6 | 86.7 | 93.8 |
| hopper-medium-expert | 52.5 | 110.9 | 107.6 | 55.8 | 103.3 | 98.0 | 105.4 | 91.5 | 110.9 |
| walker2d-medium-expert | 107.5 | 109.0 | 108.1 | 74.5 | 113.0 | 110.1 | 108.8 | 109.6 | 109.3 |
| locomotion total | 295.9 | 491.8 | 488.4 | 277.8 | 494.8 | 486.1 | 523.5 | 500.6 | 534.9 |

of the antmaze-medium and antmaze-large environments are the same. The only axis of variation in the learning setup is the noise and systematic bias added to the actions of the dataset generation policies. We therefore conclude that our algorithm is more robust to the noise and systematic bias added to the actions than distribution-constrained offline RL algorithms.

Next, we evaluate our approach on Gym locomotion tasks from the standard D4RL suite. The performance results on these tasks are illustrated in Table 3.3. Our method is competitive with BC, one-step offline RL methods [BWRB21], and multi-step distribution-constraint RL methods [KTFN21, KZTL20]. This is not surprising because in these tasks, the offline dataset contains a large number of trajectories with high returns.

In terms of total amount of compute and type of resources used, we use an internal cluster that allows for access up to 64 preemptive Nvidia RTX 2080 Ti GPUs. For each experiment of learning from an offline dataset, we use half a GPU and 3 CPU cores. Each experiment generally takes half a day to finish. We implemented our algorithms in Pytorch [PGM$^+$19] and obtained results for baselines from the publicly available implementations released by the original authors.

### 3.4.2 Ablations

We conduct three different sets of experiments to gain more insights into our algorithm. The first experiment measures the importance of having an auxiliary generator. We recall that there are two benefits to having the auxiliary generator. Firstly, without the auxiliary generator, the generator does not in general match the data distribution (Theorem 3.3.1). As such, the discriminator has an unfair advantage in learning how to distinguish between real and generated examples. Secondly, the auxiliary generator plays the role of a support player and learns to output actions that are assigned non-zero probability by the data distribution, but have low Q values. The support player allows the policy to concentrate on in-support maximization of the Q-function (Theorem 3.3.2). Table 3.4 demonstrates that having an auxiliary generator clearly leads to a performance improvement across different task families, from Gym locomotion tasks to AntMaze tasks and even dexterous manipulation tasks.

The second experiment compares the performance of the policy and the auxiliary generator on a subset of the Gym locomotion and AntMaze tasks (Table 3.5). The difference in the performance of the policy and auxiliary generator illustrates their specialization of responsibility: the policy learns to output actions that lead to good performance, while the auxiliary generator learns to model the "remainder" of the data distribution. If this "remainder" also contains good action, then the auxiliary generator will have non-trivial performance. Otherwise, the auxiliary generator will have poor performance.

In the Gym locomotion tasks, the auxiliary generator has non-trivial performance, but it is still worse than the policy. This demonstrates that: 1. By optimizing the policy to maximize the long-term return and the discriminator function, the policy can outperform the auxiliary generator, which only maximizes the discriminator function, 2. The dataset contains a large fraction of medium performance level actions contained in continuous trajectories, which the

**Table 3.4**: Ablation for training without and with auxiliary generator. The dual generator technique, which trains the auxiliary generator in addition to the policy, is crucial to obtain good performance.

| Dataset | Without | With |
|---|---|---|
| halfcheetah-medium-expert | 79.8 (3.4) | 93.8 (0.1) |
| hopper-medium-expert | 95.1 (1.6) | 110.9 (0.8) |
| antmaze-large-bias | 55.0 (2.3) | 63.9 (6.0) |
| antmaze-large-noisy | 45.1 (1.8) | 54.3 (2.0) |

**Table 3.5**: Policy vs Auxiliary Generator. The auxiliary generator has reasonable performance on the easier locomotion tasks and is significantly worse than the policy on the harder AntMaze tasks.

| Dataset | Auxiliary Generator | Policy |
|---|---|---|
| halfcheetah-medium-expert | 48.5 (2.1) | 93.8 (0.1) |
| hopper-medium-expert | 70.4 (0.9) | 110.9 (0.8) |
| antmaze-large-bias | 0.0 (0.0) | 63.9 (6.0) |
| antmaze-large-noisy | 0.0 (0.0) | 54.3 (2.0) |

auxiliary generator has learnt to output. In contrast, in the `bias` and `noisy` AntMaze tasks, the auxiliary generator fails to obtain non-zero performance while the policy has strong performance. This reflects the necessity of carefully picking a subset of the in-support actions to obtain good performance.

The third set of experiments illustrates the importance of weighing the value function in the policy objective by the probability computed by the discriminator, as described in Eq. 3.11. Doing so provides a second layer of protection against exploitation of errors in the value function by the policy. Table 3.6 illustrates that this is very important for the AntMaze tasks, which require combining optimal and sub-optimal trajectories to obtain good performance. Perhaps this is because learning from such trajectories necessitates many rounds of offline policy evaluation and improvement steps, with each round creating an opportunity for the policy to exploit the errors in the value estimates. On the other hand, the dynamic weight is less important in the Gym locomotion tasks, presumably because a significant fraction of the corresponding offline datasets has high returns and therefore incorporating sub-optimal data is less criticial to obtain high performance.

**Table 3.6**: Ablation for dynamic weighting of value function estimates in the policy objective. When learning from datasets that require combining actions across trajectories, such as the AntMaze tasks, using the dynamic weighting is vital to obtaining good performance.

| Dataset | Without | With |
|---|---|---|
| halfcheetah-medium-expert | 91.1 (1.1) | 93.8 (0.1) |
| hopper-medium-expert | 106.7 (2.9) | 110.9 (0.8) |
| antmaze-large-play | 0.0 (0.0) | 64.4 (1.7) |
| antmaze-large-diverse | 0.0 (0.0) | 74.1 (2.8) |

## 3.5 Proofs for theorems in Section 3.3.1

### 3.5.1 Proof for Theorem 3.3.1

In the following proof, we use $p_{\text{data}}$ to refer to the real data distribution, instead of $p_{\mathcal{D}}$ as in Section 3.3.1, to avoid confusion with the discriminator distribution.

We recall Theorem 3.3.1:

**Theorem 3.3.1.** *The optimal generator of Eq. 3.4 induces a distribution* $p_g^*(x) = p_{\mathcal{D}}(x)\dfrac{e^{-f(x)-\nu}}{2-e^{-f(x)-\nu}}$, *where* $\nu > 0$ *is the Lagrange multiplier that ensures that* $p_g^*(x)$ *is normalized to 1.*

The optimization problem in Eq. 3.4 is:

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim p_{\text{data}}}[\log(D(x))] + \mathbb{E}_{z\sim p(z)}[\log(1-D(G(z)))] + \mathbb{E}_{z\sim p(z)}[f(G(z))]$$

The proof proceeds as follows: We first simplify the objective function into two terms. The first term is the Jensen–Shannon divergence between the data distribution and the distribution induced by the generator [GPAM$^+$14]. The second term is the expected value of the secondary objective function $f$. We then show that the problem is convex, where strong duality holds. We then use the KKT conditions to find the functional form of the optimal solution, which gives us Theorem 3.3.1.

We only prove the statement for discrete sample space, and we let $n$ be the size of the sample space – the random variable $x$ can take on $n$ different values.

*Proof.* Since the third term in the objective function is not a function of the discriminator $D$, for $G$ fixed, the optimal discriminator of Eq. 3.4 is $D^*_G(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}$ where $p_g$ is the distribution induced by the generator $G$. (similar to Prop 1 in [GPAM$^+$14] ).

Similarly to how [GPAM$^+$14] shows that the GAN objective in Eq. 3.1 minimizes the JS divergence between the data distribution and the distribution induced by the generator, we can now rewrite the objective in Eq. 3.4 as:

$$V(G, D^*_G) \tag{3.14}$$

$$= \mathbb{E}_{x \sim p_{\text{data}}}[\log(D^*_G(x))] + \mathbb{E}_{z \sim p(z)}[\log(1 - D^*_G(G(z)))] + \mathbb{E}_{z \sim p(z)}[f(G(z))] \tag{3.15}$$

$$= 2JSD(p_{\text{data}}||p_g) + \mathbb{E}_{x \sim p_g}[f(x)] - \log 4 \tag{3.16}$$

For conciseness, let $g^{(i)} = p_g(x_i)$ be the probability that $p_g$ assigns to $x_i$ and $g = [g^{(1)}, \ldots, g^{(n)}]^T$ be a column vector containing the probabilities that $p_g$ assigns to each possible values of $x$, from $x_1$ to $x_n$.

Similarly, let $f^{(i)} = f(x_i)$ be the value that the secondary objective $f$ assigns to $x_i$. We also overload the notation to let $f = [f^{(1)}, \ldots, f^{(n)}]^T$ be a column vector containing the values that the secondary objective $f$ assigns to each possible value of the random variable $x$, from $x_1$ to $x_n$.

Also let $p^{(i)}_{\text{data}} = p_{\text{data}}(x_i)$ be the probability that the data distribution assigns to $x_i$.

We can then rewrite the problem in Eq. 3.4 in a standard form [BV04] as:

$$\min_g \quad 2JSD(p_{\text{data}}||p_g) + g^T f \tag{3.17}$$

$$\text{s.t.} \quad -g^{(i)} \leq 0 \tag{3.18}$$

$$\mathbf{1}^T g - 1 = 0 \tag{3.19}$$

where $\mathbf{1}$ is a column vector of 1, which has the same number of entries as the vector $g$. The constraint 3.18 ensures that the probability that $p_g$ assigns to any $x$ is non-negative and the constraint 3.19 ensures the probabilities sum up to 1.

The problem is convex because the objective function is a nonnegative weighted sum of two convex functions (JSD is convex because JSD is itself a nonnegative weighted sum of KL, which is a convex function).

Strong duality also holds because Slater's condition holds. A strictly feasible point for Slater's condition to hold is the uniform distribution, i.e. $g^{(i)} = \frac{1}{n}, \forall i$.

The Lagrangian is:

$$L = 2JSD(p_{\text{data}}||p_g) + g^T f - \sum_i \lambda^{(i)} g^{(i)} + \nu(\mathbf{1}^T g - 1) \tag{3.20}$$

where $\lambda^{(i)}$ and $\nu$ are the Lagrangian multipliers.

For any $i \in [1,n]$, the partial derivative of the Lagrangian with respect to $g^{(i)}$ is:

$$\frac{\partial L}{\partial g^{(i)}} = log\left(\frac{2g^{(i)}}{p_{\text{data}}^{(i)} + g^{(i)}}\right) + f^{(i)} - \lambda^{(i)} + \nu \tag{3.21}$$

Let $g_*$ and $(\lambda_*, \nu_*)$ be the primal and dual optimal solutions of the optimization problem. As the strong duality holds, the variables $g_*$ and $(\lambda_*, \nu_*)$ must satisfy the KKT conditions. For any $i \in [1,n]$, the following holds:

$$-g_*^{(i)} \leq 0 \tag{3.22}$$

$$\mathbf{1}^T g_* - 1 = 0 \tag{3.23}$$

$$\lambda_*^{(i)} \geq 0 \tag{3.24}$$

$$\lambda_*^{(i)} g_*^{(i)} = 0 \tag{3.25}$$

$$\frac{\partial L}{\partial g^{(i)}} = log\left(\frac{2g_*^{(i)}}{p_{\text{data}}^{(i)} + g_*^{(i)}}\right) + f^{(i)} - \lambda_*^{(i)} + \nu_* = 0 \tag{3.26}$$

From Equation 3.26, we have $\lambda_*^{(i)} = log\left(\frac{2g_*^{(i)}}{p_{\text{data}}^{(i)} + g_*^{(i)}}\right) + f^{(i)} + \nu_*$, and substitute into Equation 3.25:

$$\left[log\left(\frac{2g_*^{(i)}}{p_{\text{data}}^{(i)} + g_*^{(i)}}\right) + f^{(i)} + \nu_*\right] g_i^* = 0 \tag{3.27}$$

We consider what happens when $g_i^* > 0$, due to complementary slackness, we have:

$$\log\left(\frac{2g_*^{(i)}}{p_{\text{data}}^{(i)} + g_*^{(i)}}\right) + f^{(i)} + \nu_* = 0 \tag{3.28}$$

$$\implies g_*^{(i)} = \frac{p_{\text{data}}^{(i)} e^{-f^{(i)} - \nu_*}}{(2 - e^{-f^{(i)} - \nu_*})} \tag{3.29}$$

$$p_g^*(x_i) = p_{\text{data}}(x_i) \frac{e^{-f(x_i) - \nu_*}}{2 - e^{-f(x_i) - \nu_*}} \tag{3.30}$$

We can then pick an appropriate value for the Lagrange multiplier $\nu$ such that the probabilities $p_g^*(x_i)$ normalize to 1. QED.

### 3.5.2 Proof for Theorem 3.3.2

In the following proof, we use $p_{\text{data}}$ to refer to the real data distribution, instead of $p_{\mathcal{D}}$ as in Section 3.3.1, to avoid confusion with the discriminator distribution.

Recall that we define $p_{mix}$ as $p_{mix} = \frac{p_g + p_{aux}}{2}$. Theorem 3.3.2 is stated in reference to the optimization problem in Eq. 3.5, which we restate here:

$$\min_{G,G_{aux}} \max_{D} \quad V(G,G_{aux},D) = \mathbb{E}_{x\sim p_{\text{data}}}[\log(D(x))] + \mathbb{E}_{x\sim p_{mix}}[\log(1-D(x))] + \mathbb{E}_{x\sim p_g}[f(x)] \tag{3.31}$$

where the first two terms in the objective function are the GAN objective and the last term is the secondary objective function.

Similar to the proof for Theorem 3.3.1, we can rewrite the objective function in Eq. 3.31 as [GPAM$^+$14]:

$$V(G,G_{aux},D^*) \tag{3.32}$$

$$= 2JSD(p_{\text{data}}||\frac{p_g + p_{aux}}{2}) + \mathbb{E}_{x\sim p_g}[f(x)] - \log 4 \tag{3.33}$$

We are only interested in optimizing for the secondary objective function $f$ in the space of optimal GAN solutions. We therefore enforce that $p_{mix} = \frac{p_g + p_{aux}}{2} = p_{\text{data}}$, which makes the JSD term vanish in Eq. 3.33 and allows us to solve the following optimization problem.

$$\min_{G} \quad \mathbb{E}_{x\sim p_g}[f(x)] \tag{3.34}$$

$$\text{s.t.} \quad p_g \leq 2p_{\text{data}} \tag{3.35}$$

$$p_{aux} = 2p_{\text{data}} - p_g \tag{3.36}$$

We claim that the solution to the optimization problem above is as follows. We define $x_0$ to be the element inside the support of the data distribution $p_{\text{data}}$ that minimizes $f$, i.e.

$x_0 = \underset{x \in \mathrm{Supp}(p_{\mathrm{data}})}{\arg\min} \ f(x)$. The optimal primary generator $p_g^*$ assigns the following probability to $x_0$:

$$p_g^*(x_0) = \begin{cases} 2p_{\mathrm{data}}(x_0) & \text{if} \quad 2p_{\mathrm{data}}(x_0) < 1 \\ 1 & \text{otherwise} \end{cases} \tag{3.37}$$

If the global maximum $x_0$ is not taking the full probability mass, the rest of the probability mass is redistributed to the next best in-support maxima, which we can define recursively:

$$\text{For } x_i \in \underset{x \in \mathrm{Supp}(p_{\mathrm{data}}) \setminus \{x_j\}_{j=0}^{i-1}}{\arg\min} f(x),\ p_g^*(x_i) = \begin{cases} 2p_{\mathrm{data}}(x_i) & \text{if} \quad \sum_{j=0}^{i} p_g^*(x_j) < 1 \\ 1 - \sum_{j=0}^{i-1} p_g^*(x_j) & \text{if} \quad \sum_{j=0}^{i} p_g^*(x_j) > 1 \\ 0 & \text{if} \quad \sum_{j=0}^{i-1} p_g^*(x_j) = 1 \end{cases} \tag{3.38}$$

*Proof.*

We show the proof by contradiction. That is, assume that there exists another distribution $p_g^a$ with the following properties:

- There exists $x$ where $p_g^a(x) \neq p_g^*(x)$
- $p_g^a$ satisfies the constraint (3.35)-(3.36)
- The value of the objective function achieved by $p_g^a$ is better than the value achieved by $p_g^*$. That is, $\mathbb{E}_{x \sim p_g^a}[f(x)] < \mathbb{E}_{x \sim p_g^*}[f(x)]$.

We will show that the existence of such a distribution $p_g^a$ will lead to contradiction,

We separate the analyses into three different cases, depending on the property of $p_g^*$:

- Case 1: $p_g^*$ assigns all probability mass to $x_0$
- Case 2: If $p_g^*$ assigns non-zero probability to $x$, then $p_g^* = 2p_{\mathrm{data}}(x)$

- Case 3: There exists an $x$ where $2p_{\text{data}}(x) > p_g^*(x) > 0$

We will walk through the three cases independently and show the contradiction in each case.

**Case 1:** $p_g^*$ assigns the full probability mass to $x_0$, that is $p_g^*(x_0) = 1$, and assigns zero probability to every $x$ not equal to $x_0$. Without loss of generality, we consider $p_g$ that assigns non-zero probability to a $x_k \neq x_0$, assigns the remaining probability mass to $x_0$, and assigns zero probability to all $x$ that is not equal to either $x_0$ or $x_k$. That is, assume there exists $p_g^a$ such that:

$$0 > p_g^a(x_0) > 1 \tag{3.39}$$

$$p_g^a(x_k) = 1 - p_g^a(x_0) > 0 \text{ for some } x_k \in \text{Supp}(p_{\text{data}}) \tag{3.40}$$

$$\mathbb{E}_{x \sim p_g^*}[f(x)] - \mathbb{E}_{x \sim p_g^a}[f(x)] > 0 \tag{3.41}$$

where $x_k \in \text{Supp}(p_{\text{data}})$ follows from constraint 3.35 ($p_g \leq 2p_{\text{data}}$, and thus $p_g^a$ can only assign non-zero probability to $x$ within the support of $p_{\text{data}}$). We can then show that:

$$\mathbb{E}_{x \sim p_g^*}[f(x)] - \mathbb{E}_{x \sim p_g^a}[f(x)] \tag{3.42}$$

$$= f(x_0) - p_g^a(x_0)f(x_0) - p_g^a(x_k)f(x_k) \tag{3.43}$$

$$= (1 - p_g^a(x_0))f(x_0) - p_g^a(x_k)f(x_k) \tag{3.44}$$

$$= p_g^a(x_k)f(x_0) - p_g^a(x_k)f(x_k) \tag{3.45}$$

$$= p_g^a(x_k)[f(x_0) - f(x_k)] \leq 0 \text{ (contradiction with Eq.3.41)} \tag{3.46}$$

where the last inequity follows from these two facts:

$$x_0 = \underset{x \in \text{Supp}(p_{\text{data}})}{\arg\min} f(x) \tag{3.47}$$

$$x_k \in \text{Supp}(p_{\text{data}}) \tag{3.48}$$

**Case 2:**

$$p_g^*(x) = \begin{cases} 2p_{\text{data}}(x) & \text{if} \quad p_g^*(x) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.49}$$

Let $\{x_0, \ldots, x_i\}$ be the set of x where $p_g^*(x) > 0$, then we also require that $\sum_{j=0}^{i} p_g^*(x) = 1$.

Without loss of generality, we assume a distribution $p_g^a$ exists with the following properties. There exists $x_m, x_n$ such that:

$$p_g^*(x_m) = 2p_{\text{data}}(x_m) > 0 \quad \text{and} \quad p_g^a(x_m) < 2p_{\text{data}}(x_m) \tag{3.50}$$

$$p_g^*(x_n) = 0 \quad \text{and} \quad p_g^a(x_n) = 2p_{\text{data}}(x_m) - p_g^a(x_m) > 0 \tag{3.51}$$

$$p_g^*(x) = p_g^a(x) \text{ otherwise (that is, for all } x \notin \{x_m, x_n\}) \tag{3.52}$$

$$\mathbb{E}_{x \sim p_g^*}[f(x)] - \mathbb{E}_{x \sim p_g^a}[f(x)] > 0 \tag{3.53}$$

We note that $f(x_m) \leq f(x_n)$ since $p_g^*$ assigns non-zero probability to $x_m$ and assigns zero probability to $x_n$.

We can show that:

$$\begin{aligned}
&\mathbb{E}_{x\sim p_g^*}[f(x)] - \mathbb{E}_{x\sim p_g^a}[f(x)] && (3.54)\\
&\quad = p_g^*(x_m)f(x_m) - p_g^a(x_m)f(x_m) - p_g^a(x_n)f(x_n) && (3.55)\\
&\quad = p_g^*(x_m)f(x_m) - p_g^a(x_m)f(x_m) - p_g^a(x_n)f(x_n) && (3.56)\\
&\quad = p_g^*(x_m)f(x_m) - p_g^a(x_m)f(x_m) - (2p_{\text{data}}(x_m) - p_g^a(x_m))f(x_n) && (3.57)\\
&\quad = p_g^*(x_m)f(x_m) - p_g^a(x_m)f(x_m) - 2p_{\text{data}}(x_m)f(x_n) + p_g^a(x_m)f(x_n) && (3.58)\\
&\quad = p_g^*(x_m)f(x_m) - p_g^a(x_m)f(x_m) - p_g^*(x_m)f(x_n) + p_g^a(x_m)f(x_n) && (3.59)\\
&\quad = p_g^*(x_m)[f(x_m) - f(x_n)] - p_g^a(x_m)[f(x_m) - f(x_n)] && (3.60)\\
&\quad = [f(x_m) - f(x_n)][p_g^*(x_m) - p_g^a(x_m)] \leq 0 \text{ (contradiction with Eq.3.53)} && (3.61)
\end{aligned}$$

where the last inequality is true because $f(x_m) \leq f(x_n)$ as we noted above, and $p_g^*(x_m) = 2p_{\text{data}}(x_m) > p_g^a(x_m)$.

**Case 3:**

There exists $x_i$ such that $2p_{\text{data}}(x_i) > p_g^*(x_i) > 0$. For all $x \neq x_i$:

$$p_g^*(x) = \begin{cases} 2p_{\text{data}}(x) & \text{if} \quad p_g^*(x) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.62}$$

Let $\{x_0, \ldots, x_i\}$ be the set of x where $p_g^*(x) > 0$, we also require $\sum_{j=0}^{i} p_g^*(x) = 1$.

Without loss of generality, there are three cases we need to consider for the distribution $p_g^a$, each yielding a contradiction:

- $p_g^a(x_i) = p_g^*(x_i)$, but there exists $x$ such that $p_g^a(x) \neq p_g^*(x)$.
- $p_g^a(x_i) > p_g^*(x_i)$.
- $p_g^a(x_i) < p_g^*(x_i)$.

In each case, the proof by contradiction is similar to the proof in Case 2 above, where we pick a pair of $x_m, x_n$ and shows that $p_g^a$ can not achieve a lower value of the objective function than $p_g^*$. We thus do not repeat the argument here. QED

## 3.6 Conclusions

In this paper, we introduced DASCO, a GAN-based offline RL method that addresses the challenges of training policies as generators with a discriminator to minimize deviation from the behavior policy by means of two modifications: an auxiliary generator to turn the GAN loss into a support constraint, and a value function weight in the policy objective. The auxiliary generator makes it possible for the policy to focus on maximizing the value function without needing to match the *entirety* of the data distribution, only that part of it that has high value, effectively turning the standard distributional constraint that would be enforced by a conventional GAN into a kind of support constraint. This technique may in fact be of interest in other settings where there is a need to maximize some objective in addition to fooling a discriminator, and applications of this approach outside of reinforcement learning are an exciting direction for future work. Further, since our method enables GAN-based strategies to attain good results on a range of offline RL benchmark tasks, it would also be interesting in future work to consider other types of GAN losses that induce different divergence measures. We also plan to explore robust methods for offline policy and hyper-parameter selection in the future.

## 3.7 Acknowledgement

Computing needs were supported by the Nautilus Pacific Research Platform. Chapter 3, in part, is a reprint of material from Quan Vuong, Aviral Kumar, Sergey Levine, Yevgen Chebotar. “Dual Generator Offline Reinforcement Learning”. In International Conference on

Machine Learning Decision Awareness in Reinforcement Learning 2022 and In submission at Neural Information Processing Systems, 2022. The dissertation author is the primary author of the paper.

# Chapter 4

# Supervised Policy Update for Deep Reinforcement Learning

The policy gradient problem in deep reinforcement learning (DRL) can be defined as seeking a parameterized policy with high expected reward. An issue with policy gradient methods is poor sample efficiency [Kak03, SLM$^+$15, WMG$^+$17, SWD$^+$17]. In algorithms such as REINFORCE [Wil92a], new samples are needed for every gradient step. When generating samples is expensive (such as robotic environments), sample efficiency is of central concern. The sample efficiency of an algorithm is defined to be the number of calls to the environment required to attain a specified performance level [Kak03].

Thus, given the current policy and a fixed number of trajectories (samples) generated, the goal of the sample efficiency problem is to construct a new policy with the highest performance improvement possible. To do so, it is desirable to limit the search to policies that are close to the original policy $\pi_{\theta_k}$ [Kak02, SLM$^+$15, WMG$^+$17, AHTA17, SWD$^+$17, TAS18]. Intuitively, if the candidate new policy $\pi_\theta$ is far from the original policy $\pi_{\theta_k}$, it may not perform better than the original policy because too much emphasis is being placed on the relatively small batch of new data generated by $\pi_{\theta_k}$, and not enough emphasis is being placed on the relatively large amount of data and effort previously used to construct $\pi_{\theta_k}$.

This guideline of limiting the search to nearby policies seems reasonable in principle, but requires a distance $\eta(\pi_\theta, \pi_{\theta_k})$ between the current policy $\pi_{\theta_k}$ and the candidate new policy $\pi_\theta$, and then attempt to solve the constrained optimization problem:

$$\underset{\theta}{\text{maximize}} \quad \hat{J}(\pi_\theta \mid \pi_{\theta_k}, \text{new data}) \tag{4.1}$$

$$\text{subject to} \quad \eta(\pi_\theta, \pi_{\theta_k}) \leq \delta \tag{4.2}$$

where $\hat{J}(\pi_\theta \mid \pi_{\theta_k}, \text{new data})$ is an estimate of $J(\pi_\theta)$, the performance of policy $\pi_\theta$, based on the previous policy $\pi_{\theta_k}$ and the batch of fresh data generated by $\pi_{\theta_k}$. The objective equation 4.1 attempts to maximize the performance of the updated policy, and the constraint equation 4.2 ensures that the updated policy is not too far from the policy $\pi_{\theta_k}$ that was used to generate the data. Several recent papers [Kak02, SLM$^+$15, SWD$^+$17, TAS18] belong to the framework equation 4.1-equation 4.2.

We propose a new methodology, called Supervised Policy Update (SPU), for this sample efficiency problem. The methodology is general in that it applies to both discrete and continuous action spaces, and can address a wide variety of constraint types for equation 4.2. Starting with data generated by the current policy, SPU optimizes over a proximal policy space to find an optimal *non-parameterized policy*. It then solves a supervised regression problem to convert the non-parameterized policy to a parameterized policy, from which it draws new samples. We develop a general methodology for finding an optimal policy in the non-parameterized policy space, and then illustrate the methodology for three different definitions of proximity. We also show how the Natural Policy Gradient and Trust Region Policy Optimization (NPG/TRPO) problems and the Proximal Policy Optimization (PPO2017) problem can be addressed by this methodology. While SPU is substantially simpler than NPG/TRPO in terms of mathematics and implementation, our extensive experiments show that SPU is more sample efficient than TRPO in

Mujoco simulated robotic tasks and PPO2017 in Atari video game tasks. Our work also strikes the right balance between performance and simplicity. The implementation is only slightly more involved than PPO2017 [SWD+17]. Simplicity in RL algorithms has its own merits.

Off-policy RL algorithms generally achieve better sample efficiency than on-policy algorithms [HZAL18a]. However, the performance of an on-policy algorithm can usually be substantially improved by incorporating off-policy training ([MKS+15a], [WBH+16]). Our paper focuses on igniting interests in separating finding the optimal policy into a two-step process: finding the optimal non-parameterized policy, and then parameterizing this optimal policy. We also wanted to deeply understand the on-policy case before adding off-policy training. We thus compare with algorithms operating under the same algorithmic constraints, one of which is being on-policy. We leave the extension to off-policy to future work. We do not claim state-of-the-art results at the time of publication of this contribution.

## 4.1 Preliminaries

We consider a Markov Decision Process (MDP) with state space $\mathcal{S}$, action space $\mathcal{A}$, and reward function $r(s,a)$, $s \in \mathcal{S}$, $a \in \mathcal{A}$. Let $\pi = \{\pi(a|s) : s \in \mathcal{S}, a \in \mathcal{A}\}$ denote a policy, let $\Pi$ be the set of all policies, and let the expected discounted reward be:

$$J(\pi) \triangleq \mathop{\mathbb{E}}_{\tau \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] \tag{4.3}$$

where $\gamma \in (0,1)$ is a discount factor and $\tau = (s_0, a_0, s_1, \dots)$ is a sample trajectory. Let $A^{\pi}(s,a)$ be the advantage function for policy $\pi$ [Lev17]. Deep reinforcement learning considers a set of parameterized policies $\Pi_{\mathrm{DL}} = \{\pi_\theta | \theta \in \Theta\} \subset \Pi$, where each policy is parameterized by a neural network called the policy network. In this paper, we will consider optimizing over the parameterized policies in $\Pi_{\mathrm{DL}}$ as well as over the non-parameterized policies in $\Pi$. For concreteness, we assume that the state and action spaces are finite. However, our methodology also applies to continuous state and action spaces, as shown in the Appendix.

One popular approach to maximizing $J(\pi_\theta)$ over $\Pi_{\text{DL}}$ is to apply stochastic gradient ascent. The gradient of $J(\pi_\theta)$ evaluated at a specific $\theta = \theta_k$ can be shown to be [Wil92a]:

$$\nabla J(\pi_{\theta_k}) = \mathop{\mathbb{E}}_{\tau \sim \pi_{\theta_k}} \left[ \sum_{t=0}^{\infty} \gamma^t \nabla \log \pi_{\theta_k}(a_t|s_t) A^{\pi_{\theta_k}}(s_t, a_t) \right]. \tag{4.4}$$

We can approximate (4.4) by sampling N trajectories of length T from $\pi_{\theta_k}$:

$$\nabla J(\pi_{\theta_k}) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \nabla \log \pi_{\theta_k}(a_{it}|s_{it}) A^{\pi_{\theta_k}}(s_t, a_t) \triangleq g_k \tag{4.5}$$

Additionally, define $d^\pi(s) \triangleq (1-\gamma)\sum_{t=0}^{\infty} \gamma^t P_\pi(s_t = s)$ for the the future state probability distribution for policy $\pi$, and denote $\pi(\cdot|s)$ for the probability distribution over the action space $\mathcal{A}$ when in state $s$ and using policy $\pi$. Further denote $D_{\text{KL}}(\pi \parallel \pi_{\theta_k})[s]$ for the KL divergence from $\pi(\cdot|s)$ to $\pi_{\theta_k}(\cdot|s)$, and denote the following as the "aggregated KL divergence".

$$\bar{D}_{\text{KL}}(\pi \parallel \pi_{\theta_k}) = \mathop{\mathbb{E}}_{s \sim d^{\pi_{\theta_k}}} [D_{\text{KL}}(\pi \parallel \pi_{\theta_k})[s]] \tag{4.6}$$

### 4.1.1 Surrogate Objectives for the Sample Efficiency Problem

For the sample efficiency problem, the objective $J(\pi_\theta)$ is typically approximated using samples generated from $\pi_{\theta_k}$ [SLM$^+$15, AHTA17, SWD$^+$17]. Two different approaches are typically used to approximate $J(\pi_\theta) - J(\pi_{\theta_k})$. We can make a first order approximation of $J(\pi_\theta)$ around $\theta_k$ [Kak02, PS08a, PS08c, SLM$^+$15]:

$$J(\pi_\theta) - J(\pi_{\theta_k}) \approx (\theta - \theta_k)^T \nabla J(\pi_{\theta_k}) \approx (\theta - \theta_k)^T g_k \tag{4.7}$$

where $g_k$ is the sample estimate equation 4.5. The second approach is to approximate the state distribution $d^{\pi}(s)$ with $d^{\pi_{\theta_k}}(s)$ [AHTA17, SWD+17, Ach]:

$$J(\pi) - J(\pi_{\theta_k}) \approx \mathcal{L}_{\pi_{\theta_k}}(\pi) \triangleq \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{s\sim d^{\pi_{\theta_k}}} \mathop{\mathbb{E}}_{a\sim\pi_{\theta_k}(\cdot|s)} \left[ \frac{\pi(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a) \right] \tag{4.8}$$

There is a well-known bound for the approximation equation 4.8 [KL02, AHTA17]. Furthermore, the approximation $\mathcal{L}_{\pi_{\theta_k}}(\pi_\theta)$ matches $J(\pi_\theta) - J(\pi_{\theta_k})$ to the first order with respect to the parameter $\theta$ [AHTA17].

## 4.2 Related Work

Natural gradient [Ama98] was first introduced to policy gradient by Kakade [Kak02] and then in [PS08a, PS08c, Ach, SLM+15]. referred to collectively here as NPG/TRPO. Algorithmically, NPG/TRPO finds the gradient update by solving the sample efficiency problem (4.1)-(4.2) with $\eta(\pi_\theta, \pi_{\theta_k}) = \bar{D}_{\mathrm{KL}}(\pi_\theta \parallel \pi_{\theta_k})$, i.e., use the aggregate KL-divergence for the policy proximity constraint equation 4.2. NPG/TRPO addresses this problem in the parameter space $\theta \in \Theta$. First, it approximates $J(\pi_\theta)$ with the first-order approximation equation 4.7 and $\bar{D}_{\mathrm{KL}}(\pi_\theta \parallel \pi_{\theta_k})$ using a similar second-order method. Second, it uses samples from $\pi_{\theta_k}$ to form estimates of these two approximations. Third, using these estimates (which are functions of $\theta$), it solves for the optimal $\theta^*$. The optimal $\theta^*$ is a function of $g_k$ and of $h_k$, the sample average of the Hessian evaluated at $\theta_k$. TRPO also limits the magnitude of the update to ensure $\bar{D}_{\mathrm{KL}}(\pi_\theta \parallel \pi_{\theta_k}) \leq \delta$ (i.e., ensuring the sampled estimate of the aggregated KL constraint is met *without* the second-order approximation).

SPU takes a very different approach by first (i) posing and solving the optimization problem in the non-parameterized policy space, and then (ii) solving a supervised regression problem to find a parameterized policy that is near the optimal non-parameterized policy. A recent paper, Guided Actor Critic (GAC), independently proposed a similar decomposition [TAS18].

However, GAC is much more restricted in that it considers only one specific constraint criterion (aggregated reverse-KL divergence) and applies only to continuous action spaces. Furthermore, GAC incurs significantly higher computational complexity, e.g. at every update, it minimizes the dual function to obtain the dual variables using SLSQP. MPO also independently propose a similar decomposition [AST$^+$18a]. MPO uses much more complex machinery, namely, Expectation Maximization to address the DRL problem. However, MPO has only demonstrates preliminary results on problems with discrete actions whereas our approach naturally applies to problems with either discrete or continuous actions. In both GAC and MPO, working in the non-parameterized space is a by-product of applying the main ideas in those papers to DRL. Our paper demonstrates that the decomposition alone is a general and useful technique for solving constrained policy optimization.

Clipped-PPO [SWD$^+$17] takes a very different approach to TRPO. At each iteration, PPO makes many gradient steps while only using the data from $\pi_{\theta_k}$. Without the clipping, PPO is the approximation equation 4.8. The clipping is analogous to the constraint equation 4.2 in that it has the goal of keeping $\pi_\theta$ close to $\pi_{\theta_k}$. Indeed, the clipping keeps $\pi_\theta(a_t|s_t)$ from becoming neither much larger than $(1+\varepsilon)\pi_{\theta_k}(a_t|s_t)$ nor much smaller than $(1-\varepsilon)\pi_{\theta_k}(a_t|s_t)$. Thus, although the clipped PPO objective does not squarely fit into the optimization framework (4.1)-(4.2), it is quite similar in spirit. We note that the PPO paper considers adding the KL penalty to the objective function, whose gradient is similar to ours. However, this form of gradient was demonstrated to be inferior to Clipped-PPO. To the best of our knowledge, it is only until our work that such form of gradient is demonstrated to outperform Clipped-PPO.

Actor-Critic using Kronecker-Factored Trust Region (ACKTR) [WMG$^+$17] proposed using Kronecker-factored approximation curvature (K-FAC) to update both the policy gradient and critic terms, giving a more computationally efficient method of calculating the natural gradients. ACER [WBH$^+$16] exploits past episodes, linearizes the KL divergence constraint, and maintains an average policy network to enforce the KL divergence constraint. In future work, it would of interest to extend the SPU methodology to handle past episodes. In contrast to bounding the KL divergence on the action distribution as we have done in this work, Relative Entropy Policy Search considers bounding the joint distribution of state and action and was only demonstrated to work for small problems [PMA10].

## 4.3 SPU Framework

The SPU methodology has two steps. In the first step, for a given constraint criterion $\eta(\pi, \pi_{\theta_k}) \leq \delta$, we find the optimal solution to the non-parameterized problem:

$$\underset{\pi \in \Pi}{\text{maximize}} \quad \mathcal{L}_{\pi_{\theta_k}}(\pi) \tag{4.9}$$

$$\text{subject to} \quad \eta(\pi, \pi_{\theta_k}) \leq \delta \tag{4.10}$$

Note that $\pi$ is not restricted to the set of parameterized policies $\Pi_{DL}$. As commonly done, we approximate the objective function (4.8). However, unlike PPO/TRPO, we are not approximating the constraint equation 4.2. We will show below the optimal solution $\pi^*$ for the non-parameterized problem (4.9)-(4.10) can be determined nearly in closed form for many natural constraint criteria $\eta(\pi, \pi_{\theta_k}) \leq \delta$.

In the second step, we attempt to find a policy $\pi_\theta$ in the parameterized space $\Pi_{DL}$ that is close to the target policy $\pi^*$. Concretely, to advance from $\theta_k$ to $\theta_{k+1}$, we perform the following steps:

1. We first sample $N$ trajectories using policy $\pi_{\theta_k}$, giving sample data $(s_i, a_i, A_i)$, $i = 1, .., m$. Here $A_i$ is an estimate of the advantage value $A^{\pi_{\theta_k}}(s_i, a_i)$. (For simplicity, we index the samples with $i$ rather than with $(i, t)$ corresponding to the $t$th sample in the $i$th trajectory.)

2. For each $s_i$, we define the target distribution $\pi^*$ to be the optimal solution to the constrained optimization problem (4.9)-(4.10) for a specific constraint $\eta$.

3. We then fit the policy network $\pi_\theta$ to the target distributions $\pi^*(\cdot|s_i)$, $i = 1,..,m$. Specifically, to find $\theta_{k+1}$, we minimize the following supervised loss function:

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{m} D_{\mathrm{KL}}(\pi_\theta \parallel \pi^*)[s_i] \tag{4.11}$$

   For this step, we initialize with the weights for $\pi_{\theta_k}$. We minimize the loss function $L(\theta)$ with stochastic gradient descent methods. The resulting $\theta$ becomes our $\theta_{k+1}$.

## 4.4 SPU Applied to Specific Proximity Criteria

To illustrate the SPU methodology, for three different but natural types of proximity constraints, we solve the corresponding non-parameterized optimization problem and derive the resulting gradient for the SPU supervised learning problem. We also demonstrate that different constraints lead to very different but intuitive forms of the gradient update.

### 4.4.1 Forward Aggregate and Disaggregate KL Constraints

We first consider constraint criteria of the form:

$$\underset{\pi\in\Pi}{\text{maximize}} \quad \sum_s d^{\pi_{\theta_k}}(s) \underset{a\sim\pi(\cdot|s)}{\mathbb{E}}\left[A^{\pi_{\theta_k}}(s,a)\right] \tag{4.12}$$

$$\text{subject to} \quad \sum_s d^{\pi_{\theta_k}}(s) D_{\mathrm{KL}}(\pi \parallel \pi_{\theta_k})[s] \le \delta \tag{4.13}$$

$$D_{\mathrm{KL}}(\pi \parallel \pi_{\theta_k})[s] \le \epsilon \text{ for all } s \tag{4.14}$$

Note that this problem is equivalent to minimizing $\mathcal{L}_{\pi_{\theta_k}}(\pi)$ subject to the constraints equation 4.13 and equation 4.14. We refer to equation 4.13 as the ”aggregated KL constraint” and to equation 4.14 as the ”disaggregated KL constraint”. These two constraints taken together restrict $\pi$ from deviating too much from $\pi_{\theta_k}$. We shall refer to equation 4.12-equation 4.14 as the forward-KL non-parameterized optimization problem.

Note that this problem without the disaggregated constraints is analogous to the TRPO problem. The TRPO paper actually prefers enforcing the disaggregated constraint to enforcing the aggregated constraints. However, for mathematical conveniences, they worked with the aggregated constraints: ”While it is motivated by the theory, this problem is impractical to solve due to the large number of constraints. Instead, we can use a heuristic approximation which considers the average KL divergence” [SLM$^+$15]. The SPU framework allows us to solve the optimization problem with the disaggregated constraints exactly. Experimentally, we compared against TRPO in a controlled experimental setting, e.g. using the same advantage estimation scheme, etc. Since we clearly outperform TRPO, we argue that SPU's two-process procedure has significant potentials.

For each $\lambda > 0$, define: $\pi^{\lambda}(a|s) = \dfrac{\pi_{\theta_k}(a|s)}{Z_{\lambda}(s)} e^{A^{\pi_{\theta_k}}(s,a)/\lambda}$ where $Z_{\lambda}(s)$ is the normalization term. Note that $\pi^{\lambda}(a|s)$ is a function of $\lambda$. Further, for each s, let $\lambda_s$ be such that $D_{\text{KL}}(\pi^{\lambda_s} \parallel \pi_{\theta_k})[s] = \epsilon$. Also let $\Gamma^{\lambda} = \{s : D_{\text{KL}}(\pi^{\lambda} \parallel \pi_{\theta_k})[s] \leq \epsilon\}$.

**Theorem 1.** *The optimal solution to the problem equation 4.12-equation 4.14 is given by:*

$$\tilde{\pi}^{\lambda}(a|s) = \begin{cases} \pi^{\lambda}(a|s) & s \in \Gamma^{\lambda} \\ \pi^{\lambda_s}(a|s) & s \notin \Gamma^{\lambda} \end{cases} \tag{4.15}$$

*where $\lambda$ is chosen so that $\sum_s d^{\pi_{\theta_k}}(s) D_{KL}(\tilde{\pi}^{\lambda} \parallel \pi_{\theta_k})[s] = \delta$ (Proof in subsection 4.6.1).*

Equation equation 4.15 provides the structure of the optimal non-parameterized policy. As part of the SPU framework, we then seek a parameterized policy $\pi_{\theta}$ that is close to $\tilde{\pi}^{\lambda}(a|s)$, that is, minimizes the loss function (4.11). For each sampled state $s_i$, a straightforward calculation shows:

$$\nabla_{\theta} D_{\text{KL}}(\pi_{\theta} \parallel \tilde{\pi}^{\lambda})[s_i] = \nabla_{\theta} D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\theta_k})[s_i] - \frac{1}{\tilde{\lambda}_{s_i}} \mathop{\mathbb{E}}_{a \sim \pi_{\theta_k}(\cdot|s_i)} [\nabla_{\theta} \log \pi_{\theta}(a|s_i) A^{\pi_{\theta_k}}(s_i, a)] \tag{4.16}$$

where $\widetilde{\lambda}_{s_i} = \lambda$ for $s_i \in \Gamma^{\lambda}$ and $\tilde{\lambda}_{s_i} = \lambda_{s_i}$ for $s_i \notin \Gamma^{\lambda}$. We estimate the expectation in equation 4.16 with the sampled action $a_i$ and approximate $A^{\pi_{\theta_k}}(s_i, a_i)$ as $A_i$ (obtained from the critic network), giving:

$$\nabla_{\theta} D_{\mathrm{KL}}(\pi_{\theta} \parallel \tilde{\pi}^{\lambda})[s_i] \approx \nabla_{\theta} D_{\mathrm{KL}}(\pi_{\theta} \parallel \pi_{\theta_k})[s_i] - \frac{1}{\widetilde{\lambda}_{s_i}} \frac{\nabla_{\theta}\pi_{\theta}(a_i|s_i)}{\pi_{\theta_k}(a_i|s_i)} A_i \tag{4.17}$$

To simplify the algorithm, we slightly modify (4.17). We replace the hyper-parameter $\delta$ with the hyper-parameter $\lambda$ and tune $\lambda$ rather than $\delta$. Further, we set $\widetilde{\lambda}_{s_i} = \lambda$ for all $s_i$ in equation 4.17 and introduce per-state acceptance to enforce the disaggregated constraints, giving the approximate gradient:

$$\nabla_{\theta} D_{\mathrm{KL}}(\pi_{\theta} \parallel \tilde{\pi}^{\lambda}) \approx \frac{1}{m} \sum_{i=1}^{m} [\nabla_{\theta} D_{\mathrm{KL}}(\pi_{\theta} \parallel \pi_{\theta_k})[s_i] - \frac{1}{\lambda} \frac{\nabla_{\theta}\pi_{\theta}(a_i|s_i)}{\pi_{\theta_k}(a_i|s_i)} A_i] \mathbb{1}_{D_{\mathrm{KL}}(\pi_{\theta} \| \pi_{\theta_k})[s_i] \leq \epsilon} \tag{4.18}$$

We make the approximation that the disaggregated constraints are only enforced on the states in the sampled trajectories. We use (4.18) as our gradient for supervised training of the policy network. The equation (4.18) has an intuitive interpretation: the gradient represents a trade-off between the approximate performance of $\pi_{\theta}$ (as captured by $\frac{1}{\lambda} \frac{\nabla_{\theta}\pi_{\theta}(a_i|s_i)}{\pi_{\theta_k}(a_i|s_i)} A_i$) and how far $\pi_{\theta}$ diverges from $\pi_{\theta_k}$ (as captured by $\nabla_{\theta} D_{\mathrm{KL}}(\pi_{\theta} \parallel \pi_{\theta_k})[s_i]$). For the stopping criterion, we train until $\frac{1}{m} \sum_i D_{\mathrm{KL}}(\pi_{\theta} \parallel \pi_{\theta_k})[s_i] \approx \delta$.

### 4.4.2 Backward KL Constraint

In a similar manner, we can derive the structure of the optimal policy when using the reverse KL-divergence as the constraint. For simplicity, we provide the result for when there are only disaggregated constraints. We seek to find the non-parameterized optimal policy by solving:

$$\underset{\pi\in\Pi}{\text{maximize}} \quad \sum_s d^{\pi_{\theta_k}}(s) \underset{a\sim\pi(\cdot|s)}{\mathbb{E}} \left[A^{\pi_{\theta_k}}(s,a)\right] \tag{4.19}$$

$$D_{\mathrm{KL}}(\pi \,\|\, \pi_{\theta_k})[s] \leq \varepsilon \text{ for all } s \tag{4.20}$$

**Theorem 2.** *The optimal solution to the problem equation 4.19-equation 4.20 is given by:*

$$\pi^*(a|s) = \pi_{\theta_k}(a|s)\frac{\lambda(s)}{\lambda'(s) - A^{\pi_{\theta_k}}(s,a)} \tag{4.21}$$

*where $\lambda(s) > 0$ and $\lambda'(s) > \max_a A^{\pi_{\theta_k}}(s,a)$ (Proof in subsection 4.6.2).*

Note that the structure of the optimal policy with the backward KL constraint is quite different from that with the forward KL constraint. A straight forward calculation shows:

$$\nabla_\theta D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi^*)[s] = \nabla_\theta D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\theta_k})[s] - \underset{a\sim\pi_{\theta_k}}{E}\left[\frac{\nabla_\theta \pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} \log\left(\frac{1}{\lambda'(s) - A^{\pi_{\theta_k}}(s,a)}\right)\right] \tag{4.22}$$

The equation (4.22) has an intuitive interpretation. It increases the probability of action $a$ if $A^{\pi_{\theta_k}}(s,a) > \lambda'(s) - 1$ and decreases the probability of action $a$ if $A^{\pi_{\theta_k}}(s,a) < \lambda'(s) - 1$. (4.22) also tries to keep $\pi_\theta$ close to $\pi_{\theta_k}$ by minimizing their KL divergence.

### 4.4.3 $L^\infty$ Constraint

In this section we show how a PPO-like objective can be formulated in the context of SPU. Recall from Section 4.2 that the the clipping in PPO can be seen as an attempt at keeping $\pi_\theta(a_i|s_i)$ from becoming neither much larger than $(1+\varepsilon)\pi_{\theta_k}(a_i|s_i)$ nor much smaller than $(1-\varepsilon)\pi_{\theta_k}(a_i|s_i)$

for $i = 1, \ldots, m$. In this subsection, we consider the constraint function

$$\eta(\pi, \pi_{\theta_k}) = \max_{i=1,\ldots,m} \frac{|\pi(a_i|s_i) - \pi_{\theta_k}(a_i|s_i)|}{\pi_{\theta_k}(a_i|s_i)} \tag{4.23}$$

which leads us to the following optimization problem:

$$\underset{\pi(a_1|s_1),\ldots,\pi(a_m|s_m)}{\text{maximize}} \quad \sum_{i=1}^{m} A^{\pi_{\theta_k}}(s_i, a_i) \frac{\pi(a_i|s_i)}{\pi_k(a_i|s_i)} \tag{4.24}$$

$$\text{subject to} \quad \left| \frac{\pi(a_i|s_i) - \pi_{\theta_k}(a_i|s_i)}{\pi_{\theta_k}(a_i|s_i)} \right| \leq \varepsilon \quad i = 1, \ldots, m \tag{4.25}$$

$$\sum_{i=1}^{m} \left( \frac{\pi(a_i|s_i) - \pi_{\theta_k}(a_i|s_i)}{\pi_{\theta_k}(a_i|s_i)} \right)^2 \leq \delta \tag{4.26}$$

Note that here we are using a variation of the SPU methodology described in Section 4.3 since here we first create estimates of the expectations in the objective and constraints and then solve the optimization problem (rather than first solve the optimization problem and then take samples as done for Theorems 1 and 2). Note that we have also included an aggregated constraint (4.26) in addition to the PPO-like constraint (4.25), which further ensures that the updated policy is close to $\pi_{\theta_k}$.

**Theorem 3.** *The optimal solution to the optimization problem (4.24-4.26) is given by:*

$$\pi^*(a_i|s_i) = \begin{cases} \pi_{\theta_k}(a_i|s_i) \min\{1 + \lambda A_i, 1 + \varepsilon\} & A_i \geq 0 \\ \pi_{\theta_k}(a_i|s_i) \max\{1 + \lambda A_i, 1 - \varepsilon\} & A_i < 0 \end{cases} \tag{4.27}$$

*for some* $\lambda > 0$ *where* $A_i \triangleq A^{\pi_{\theta_k}}(s_i, a_i)$ *(Proof in subsection 4.6.3).*

To simplify the algorithm, we treat $\lambda$ as a hyper-parameter rather than $\delta$. After solving for $\pi^*$, we seek a parameterized policy $\pi_\theta$ that is close to $\pi^*$ by minimizing their mean square error over sampled states and actions, i.e. by updating $\theta$ in the negative direction of $\nabla_\theta \sum_i (\pi_\theta(a_i|s_i) - \pi^*(a_i|s_i))^2$. This loss is used for supervised training instead of the KL because we take estimates before forming the optimization problem. Thus, the optimal values for the decision variables do not completely characterize a distribution. We refer to this approach as SPU with the $L^\infty$ constraint.

Although we consider three classes of proximity constraint, there may be yet another class that leads to even better performance. The methodology allows researchers to explore other proximity constraints in the future.

## 4.5 Algorithmic Description for SPU

## 4.6 Proofs for non-parameterized optimization problems

### 4.6.1 Forward KL Aggregated and Disaggregated Constraints

We first show that equation 4.12-equation 4.14 is a convex optimization. To this end, first note that the objective equation 4.12 is a linear function of the decision variables $\pi = \{\pi(a|s)$ $\because \quad s \in \mathcal{S},\ a \in \mathcal{A}\}$. The LHS of equation 4.14 can be rewritten as: $\sum_{a\in\mathcal{A}} \pi(a|s) \log \pi(a|s) - \sum_{a\in\mathcal{A}} \pi(a|s) \log \pi_{\theta_k}(a|s)$. The second term is a linear function of $\pi$. The first term is a convex function since the second derivative of each summand is always positive. The LHS of equation 4.14 is thus a convex function. By extension, the LHS of equation 4.13 is also a convex function since it is a nonnegative weighted sum of convex functions. The problem equation 4.12-equation 4.14 is thus a convex optimization problem. According to Slater's constraint qualification, strong duality holds since $\pi_{\theta_k}$ is a feasible solution to equation 4.12-equation 4.14 where the inequality holds strictly.

We can therefore solve equation 4.12-equation 4.14 by solving the related Lagrangian

**Algorithm 3** Algorithmic description of forward-KL non-parameterized SPU

```
Require: A neural net $\pi_\theta$ that parameterizes the policy.
Require: A neural net $V_\phi$ that approximates $V^{\pi_\theta}$.
Require: General hyperparameters: $\gamma, \beta$ (advantage estimation using GAE), $\alpha$ (learning rate), N (number of trajectory per iteration), T (size of each trajectory), M (size of training minibatch).
Require: Algorithm-specific hyperparameters: $\delta$ (aggregated KL constraint), $\epsilon$ (disaggregated constraint), $\lambda$, $\zeta$ (max number of epoch).
 1: for k = 1, 2, ... do
 2:     under policy $\pi_{\theta_k}$, sample N trajectories, each of size T $(s_{it}, a_{it}, r_{it}, s_{i(t+1)}), \quad i = 1, \dots, N, t = 1, \dots, T$
 3:     Using any advantage value estimation scheme, estimate $A_{it}, \quad i = 1, \dots, N, t = 1, \dots, T$
 4:     $\theta \leftarrow \theta_k$
 5:     $\phi \leftarrow \phi_k$
 6:     for $\zeta$ epochs do
 7:         Sample M samples from the N trajectories, giving $\{s_1, a_1, A_1, \dots, s_M, a_M, A_M\}$
 8:         $L(\phi) = \frac{1}{M}\sum_m (V^{targ}(s_m) - V_\phi(s_m))^2$
 9:         $\phi \leftarrow \phi - \alpha \nabla_\phi L(\phi)$
10:         $L(\theta) = \frac{1}{M}\sum_m \left[ \nabla_\theta D_{\mathrm{KL}}(\pi_\theta \parallel \pi_{\theta_k})[s_m] - \frac{1}{\lambda} \frac{\nabla_\theta \pi_\theta(a_m|s_m)}{\pi_{\theta_k}(a_m|s_m)} A_m \right] \mathbb{1}_{D_{\mathrm{KL}}(\pi_\theta \| \pi_{\theta_k})[s_m] \leq \epsilon}$
11:         $\theta \leftarrow \theta - \alpha L(\theta)$
12:         if $\frac{1}{m}\sum_m D_{\mathrm{KL}}(\pi \parallel \pi_{\theta_k})[s_m] > \delta$ then
13:             Break out of for loop
14:         end if
15:     end for
16:     $\theta_{k+1} \leftarrow \theta$
17:     $\phi_{k+1} \leftarrow \phi$
18: end for
```

problem. For a fixed $\lambda$ consider:

$$\underset{\pi \in \Pi}{\text{maximize}} \quad \sum_s d^{\pi_{\theta_k}}(s)\{ \underset{a \sim \pi(\cdot|s)}{\mathbb{E}}[A^{\pi_{\theta_k}}(s,a)] - \lambda D_{\mathrm{KL}}(\pi \parallel \pi_{\theta_k})[s]\} \tag{4.28}$$

$$\text{subject to} \quad D_{\mathrm{KL}}(\pi \parallel \pi_{\theta_k})[s] \leq \epsilon \text{ for all } s \tag{4.29}$$

The above problem decomposes into separate problems, one for each state $s$:

$$\underset{\pi(\cdot|s)}{\text{maximize}} \quad \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}[A^{\pi_{\theta_k}}(s,a)] - \lambda D_{\mathrm{KL}}(\pi \parallel \pi_{\theta_k})[s] \tag{4.30}$$

$$\text{subject to} \quad D_{\mathrm{KL}}(\pi \parallel \pi_{\theta_k})[s] \leq \varepsilon \tag{4.31}$$

Further consider the unconstrained problem equation 4.30 without the constraint equation 4.31:

$$\underset{\pi(\cdot|s)}{\text{maximize}} \quad \sum_{a=1}^{K} \pi(a|s)\left[A^{\pi_{\theta_k}}(s,a) - \lambda \log\left(\frac{\pi(a|s)}{\pi_{\theta_k}(a|s)}\right)\right] \tag{4.32}$$

$$\text{subject to} \quad \sum_{a=1}^{K} \pi(a|s) = 1 \tag{4.33}$$

$$\pi(a|s) \geq 0, \quad a = 1, \ldots, K \tag{4.34}$$

A simple Lagrange-multiplier argument shows that the opimal solution to (4.32)-(4.34) is given by:

$$\pi^{\lambda}(a|s) = \frac{\pi_{\theta_k}(a|s)}{Z_{\lambda}(s)} e^{A^{\pi_{\theta_k}}(s,a)/\lambda}$$

where $Z_{\lambda}(s)$ is defined so that $\pi^{\lambda}(\cdot|s)$ is a valid distribution. Now returning to the decomposed constrained problem (4.30)-(4.31), there are two cases to consider. The first case is when $D_{\mathrm{KL}}(\pi^{\lambda} \parallel \pi_{\theta_k})[s] \leq \varepsilon$. In this case, the optimal solution to (4.30)-(4.31) is $\pi^{\lambda}(a|s)$. The second case is when $D_{\mathrm{KL}}(\pi^{\lambda} \parallel \pi_{\theta_k})[s] > \varepsilon$. In this case the optimal is $\pi^{\lambda}(a|s)$ with $\lambda$ replaced with $\lambda_s$, where $\lambda_s$ is the solution to $D_{\mathrm{KL}}(\pi^{\lambda} \parallel \pi_{\theta_k})[s] = \varepsilon$. Thus, an optimal solution to equation 4.30-equation 4.31 is given by:

$$\tilde{\pi}^{\lambda}(a|s) = \begin{cases} \dfrac{\pi_{\theta_k}(a|s)}{Z(s)} e^{A^{\pi_{\theta_k}}(s,a)/\lambda} & s \in \Gamma^{\lambda} \\ \dfrac{\pi_{\theta_k}(a|s)}{Z(s)} e^{A^{\pi_{\theta_k}}(s,a)/\lambda_s} & s \notin \Gamma^{\lambda} \end{cases} \tag{4.35}$$

where $\Gamma^{\lambda} = \{s : D_{\mathrm{KL}}(\pi^{\lambda} \parallel \pi_{\theta_k})[s] \leq \varepsilon\}$.

To find the Lagrange multiplier $\lambda$, we can then do a line search to find the $\lambda$ that satisfies:

$$\sum_s d^{\pi_{\theta_k}}(s) D_{\mathrm{KL}}(\tilde{\pi}^{\lambda} \parallel \pi_{\theta_k})[s] = \delta \tag{4.36}$$

$\square$

### 4.6.2 Backward KL Constraint

The problem equation 4.19-equation 4.20 decomposes into separate problems, one for each state $s \in \mathcal{S}$:

$$\begin{aligned} \underset{\pi(\cdot|s)}{\text{maximize}} \quad & \mathop{\mathbb{E}}_{a \sim \pi_{\theta_k}(\cdot|s)} \left[ \frac{\pi(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a) \right] & (4.37) \\ \text{subject to} \quad & \mathop{\mathbb{E}}_{a \sim \pi_{\theta_k}(\cdot|s)} \left[ \log \frac{\pi_{\theta_k}(a|s)}{\pi(a|s)} \right] \leq \varepsilon & (4.38) \end{aligned}$$

After some algebra, we see that above optimization problem is equivalent to:

$$\begin{aligned} \underset{\pi(\cdot|s)}{\text{maximize}} \quad & \sum_{a=1}^{K} A^{\pi_{\theta_k}}(s,a)\pi(a|s) & (4.39) \\ \text{subject to} \quad & -\sum_{a=1}^{K} \pi_{\theta_k}(a|s) \log \pi(a|s) \leq \varepsilon' & (4.40) \\ & \sum_{a=1}^{K} \pi(a|s) = 1 & (4.41) \\ & \pi(a|s) \geq 0, \quad a = 1, \ldots, K & (4.42) \end{aligned}$$

where $\varepsilon' = \varepsilon + entropy(\pi_{\theta_k})$. equation 4.39-equation 4.42 is a convex optimization problem with Slater's condition holding. Strong duality thus holds for the problem equation 4.39-equation 4.42. Applying standard Lagrange multiplier arguments, it is easily seen that the solution to (4.39)-(4.42) is

$$\pi^*(a|s) = \pi_{\theta_k}(a|s)\frac{\lambda(s)}{\lambda'(s) - A^{\pi_{\theta_k}}(s,a)}$$

where $\lambda(s)$ and $\lambda'(s)$ are constants chosen such that the disaggregegated KL constraint is binding and the sum of the probabilities equals 1. It is easily seen $\lambda(s) > 0$ and $\lambda'(s) > \max_a A^{\pi_{\theta_k}}(s,a)$ □

### 4.6.3 $L^\infty$ constraint

The problem (4.24-4.26) is equivalent to:

$$\begin{aligned}
&\underset{\pi(a_1|s_1),\ldots,\pi(a_m|s_m)}{\text{maximize}} && \sum_{i=1}^{m} A^{\pi_{\theta_k}}(s_i,a_i)\frac{\pi(a_i|s_i)}{\pi_k(a_i|s_i)} && (4.43)\\
&\text{subject to} && 1-\varepsilon \le \frac{\pi(a_i|s_i)}{\pi_{\theta_k}(a_i|s_i)} \le 1+\varepsilon \quad i = 1,\ldots,m && (4.44)\\
& && \sum_{i=1}^{m}\left(\frac{\pi(a_i|s_i) - \pi_{\theta_k}(a_i|s_i)}{\pi_{\theta_k}(a_i|s_i)}\right)^2 \le \delta && (4.45)
\end{aligned}$$

This problem is clearly convex. $\pi_{\theta_k}(a_i|s_i), i = 1,\ldots,m$ is a feasible solution where the inequality constraint holds strictly. Strong duality thus holds according to Slater's constraint qualification. To solve equation 4.43-equation 4.45, we can therefore solve the related Lagrangian problem for fixed $\lambda$:

$$\begin{aligned}
&\underset{\pi(a_1|s_1),\ldots,\pi(a_m|s_m)}{\text{maximize}} && \sum_{i=1}^{m}\left[A^{\pi_{\theta_k}}(s_i,a_i)\frac{\pi(a_i|s_i)}{\pi_k(a_i|s_i)} - \lambda\left(\frac{\pi(a_i|s_i) - \pi_{\theta_k}(a_i|s_i)}{\pi_{\theta_k}(a_i|s_i)}\right)^2\right] && (4.46)\\
&\text{subject to} && 1-\varepsilon \le \frac{\pi(a_i|s_i)}{\pi_{\theta_k}(a_i|s_i)} \le 1+\varepsilon \quad i = 1,\ldots,m && (4.47)
\end{aligned}$$

which is separable and decomposes into m separate problems, one for each $s_i$:

$$\underset{\pi(a_i|s_i)}{\text{maximize}} \quad A^{\pi_{\theta_k}}(s_i,a_i)\frac{\pi(a_i|s_i)}{\pi_k(a_i|s_i)} - \lambda\left(\frac{\pi(a_i|s_i)-\pi_{\theta_k}(a_i|s_i)}{\pi_{\theta_k}(a_i|s_i)}\right)^2 \tag{4.48}$$

$$\text{subject to} \quad 1-\varepsilon \le \frac{\pi(a_i|s_i)}{\pi_{\theta_k}(a_i|s_i)} \le 1+\varepsilon \tag{4.49}$$

The solution to the unconstrained problem equation 4.48 without the constraint equation 4.49 is:

$$\pi^*(a_i|s_i) = \pi_{\theta_k}(a_i|s_i)\left(1+\frac{A^{\pi_{\theta_k}}(s_i,a_i)}{2\lambda}\right)$$

Now consider the contrained problem equation 4.48-equation 4.49. If $A^{\pi_{\theta_k}}(s_i,a_i) \geq 0$ and $\pi^*(a_i|s_i) > \pi_{\theta_k}(a_i|s_i)(1+\varepsilon)$, it follows that the optimal solution is $\pi_{\theta_k}(a_i|s_i)(1+\varepsilon)$. Similarly, If $A^{\pi_{\theta_k}}(s_i,a_i) < 0$ and $\pi^*(a_i|s_i) < \pi_{\theta_k}(a_i|s_i)(1-\varepsilon)$, then the optimal solution is $\pi_{\theta_k}(a_i|s_i)(1-\varepsilon)$. Rearranging the terms give us Theorem 3. To determine the value of $\lambda$, we can perform a line search over $\lambda$ so that the constraint equation 4.45 is binding. □

## 4.7 Experimental Results

We provide extensive experimental results which demonstrate SPU outperforms recent state-of-the-art methods for environments with continuous or discrete action spaces. We also provide ablation studies to show the importance of the different algorithmic components, and a sensitivity analysis to show that SPU's performance is relatively insensitive to hyper-parameter choices. There are two definitions of superior sample complexity performance that we use to compare two different algorithms A and B: (i) A is better than B if A takes fewer interactions with the environment to achieve a pre-defined performance threshold [Kak03]; (ii) A is better than B if the averaged final performance of A is higher than that of B given the same number of interactions with the environment[SWD$^+$17]. Implementation details are provided in Appendix section 4.8.

### 4.7.1 Results on Mujoco

The Mujoco [TET12] simulated robotics environments provided by OpenAI gym [BCP+16] have become a popular benchmark for control problems with continuous action spaces. In terms of final performance averaged over all available ten Mujoco environments and ten different seeds, SPU with $L^\infty$ constraint (Section 5.3) and SPU with forward KL constraints (Section 5.1) outperform TRPO by 6% and 27% respectively. Since the forward-KL approach is our best performing approach, we focus subsequent analysis on it and hereafter refer to it as SPU. Figure 4.1 illustrates the performance of SPU versus TRPO for each of the ten Mujoco environments. We observe that SPU clearly outperforms TRPO throughout training in the upper 5 environments while being roughly the same as TRPO in the lower 5. We also note that SPU significantly outperforms TRPO on the 3 most challenging Mujoco environments, which are listed in the top left corner in Figure 4.1. Algorithm 3 in the Appendix provides the complete description of the SPU algorithm.

### 4.7.2 Ablation Studies for Mujoco

We refer to the indicator variable in equation 4.18 as *per-state acceptance*. Removing this component is equivalent to removing the indicator variable. We refer to using $\sum_i D_{\mathrm{KL}}(\pi_\theta \parallel \pi_{\theta_k})[s_i]$ to determine the number of training epochs as *dynamic stopping*. Without this component, the number of training epochs is treated as a hyper-parameter. We also tried removing $\nabla_\theta D_{\mathrm{KL}}(\pi_\theta \parallel \pi_{\theta_k})[s_i]$ from the gradient update step in equation 4.18. Table 4.1 illustrates the contribution of the different components of SPU to the overall performance. The third row shows that the term $\nabla_\theta D_{\mathrm{KL}}(\pi_\theta \parallel \pi_{\theta_k})[s_i]$ makes a crucially important contribution to SPU. Furthermore, per-state acceptance and dynamic stopping are both also important for obtaining high performance, with the former playing a more central role. When a component is removed, the hyper-parameters are retuned to ensure that the best possible performance is obtained with the alternative (simpler) algorithm.

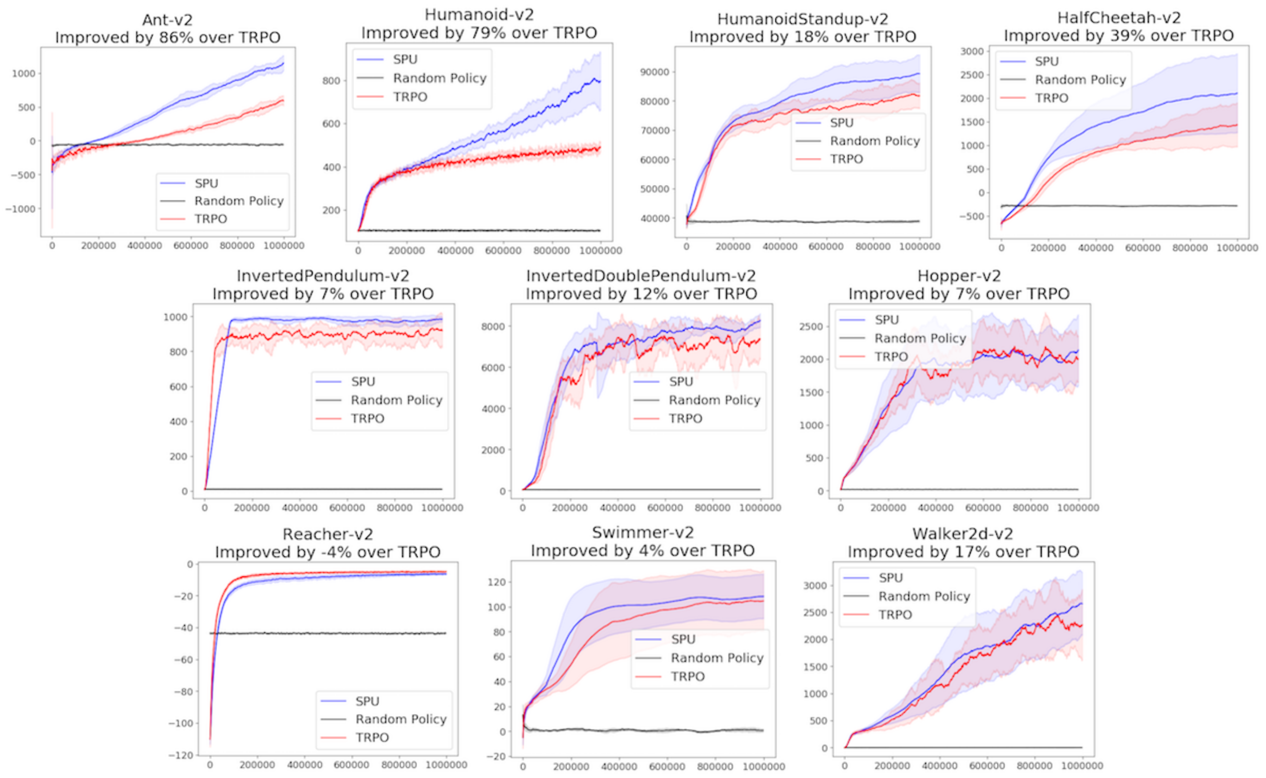


**Figure 4.1**: Performance of SPU versus TRPO on 10 Mujoco environments. The x-axis indicates timesteps. The y-axis indicates the average episode reward of the last 100 episodes.

### 4.7.3 Results on Atari

It has recently been observed that deep neural networks is not needed to obtain high performance in many Mujoco environments [RLTK17a]. To more conclusively evaluate SPU, we compare it against state-of-the-art method on the Arcade Learning Environments [BNVB12] exposed through OpenAI gym [BCP$^+$16]. Using the same network architecture and hyper-parameters, we learn to play 60 Atari games from raw pixel observations and rewards. This is highly challenging because of the diversity in the games and the high dimensionality of the observations.

Here, we compare SPU against PPO because PPO outperforms TRPO by 9% in Mujoco. Averaged over 60 Atari environments and 20 seeds, SPU is **55%** better than PPO in terms of averaged final performance. Figure 4.2 provides a high-level overview of the result. The dots in

**Table 4.1**: Ablation study for SPU

| Approach | Percentage better than TRPO | Performance vs. original algorithm |
|---|---|---|
| Original Algorithm | 27% | 0% |
| No grad KL on line 5 | 4% | - 85% |
| No dynamic stopping | 24% | - 11% |
| No per-state acceptance | 9% | - 67% |

the shaded area represent environments where their performances are roughly similar. The dots to the right of the shaded area represent environment where SPU is more sample efficient than PPO. We can draw two conclusions: (i) In 36 environments, SPU and PPO perform roughly the same ; SPU clearly outperforms PPO in 15 environments while PPO clearly outperforms SPU in 9; (ii) In those 15+9 environments, the extent to which SPU outperforms PPO is much larger than the extent to which PPO outperforms SPU. SPU's high performance in both the Mujoco and Atari domains demonstrates its high performance and generality.

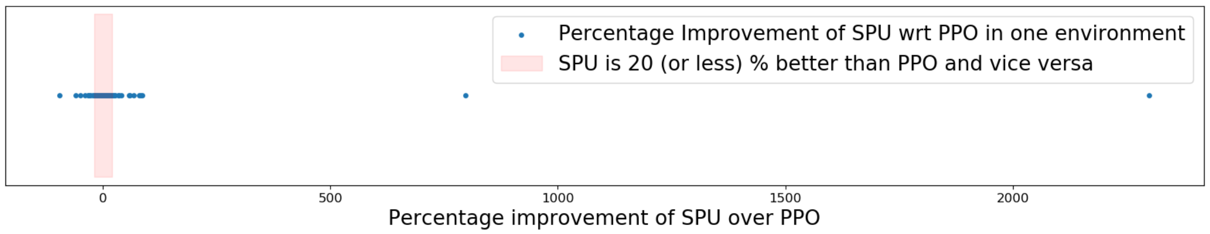


**Figure 4.2**: High-level overview of results on Atari

## 4.8 Implementation Details and Hyperparameters

### 4.8.1 Mujoco

As in [SWD+17], for Mujoco environments, the policy is parameterized by a fully-connected feed-forward neural network with two hidden layers, each with 64 units and tanh nonlinearities. The policy outputs the mean of a Gaussian distribution with state-independent variable standard deviations, following [SLM+15, DCH+16]. The action dimensions are assumed to be independent. The probability of an action is given by the multivariate Gaussian probability distribution function. The baseline used in the advantage value calculation is parameterized by a similarly sized neural network, trained to minimize the MSE between the sampled states TD$-\lambda$ returns and the their predicted values. For both the policy and baseline network, SPU and TRPO use the same architecture. To calculate the advantage values, we use Generalized Advantage Estimation [SML+15]. States are normalized by dividing the running mean and dividing by the running standard deviation before being fed to any neural networks. The advantage values are normalized by dividing the batch mean and dividing by the batch standard deviation before being used for policy update. The TRPO result is obtained by running the TRPO implementation provided by OpenAI [DHK+17], commit 3cc7df060800a45890908045b79821a13c4babdb. At every iteration, SPU collects 2048 samples before updating the policy and the baseline network. For both networks, gradient descent is performed using Adam [KB14] with step size 0.0003, minibatch size of 64. The step size is linearly annealed to 0 over the course of training. $\gamma$ and $\lambda$ for GAE [SML+15] are set to 0.99 and 0.95 respectively. For SPU, $\delta, \epsilon, \lambda$ and the maximum number of epochs per iteration are set to $0.05/1.2$, 0.05, 1.3 and 30 respectively. Training is performed for 1 million timesteps for both SPU and PPO. In the sensitivity analysis, the ranges of values for the hyper-parameters $\delta, \epsilon, \lambda$ and maximum number of epochs are $[0.05, 0.07]$, $[0.01, 0.07]$, $[1.0, 1.2]$ and $[5, 30]$ respectively.

### 4.8.2 Atari

Unless otherwise mentioned, the hyper-parameter values are the same as in subsection 4.8.1. The policy is parameterized by a convolutional neural network with the same architecture as described in [MKS+15a]. The output of the network is passed through a relu, linear and softmax layer in that order to give the action distribution. The output of the network is also passed through a different linear layer to give the baseline value. States are normalized by dividing by 255 before being fed into any network. The TRPO result is obtained by running the PPO implementation provided by OpenAI [DHK$^+$17], commit 3cc7df060800a45890908045b79821a13c4babdb. 8 different processes run in parallel to collect timesteps. At every iteration, each process collects 256 samples before updating the policy and the baseline network. Each process calculates its own update to the network's parameters and the updates are averaged over all processes before being used to update the network's parameters. Gradient descent is performed using Adam [KB14] with step size 0.0001. In each process, random number generators are initialized with a different seed according to the formula $process\_seed = experiment\_seed + 10000 * process\_rank$. Training is performed for 10 million timesteps for both SPU and PPO. For SPU, $\delta, \varepsilon, \lambda$ and the maximum number of epochs per iteration are set to 0.02, $\delta/1.3$, 1.1 and 9 respectively.

## 4.9 Acknowledgement

We would like to acknowledge the extremely helpful support by the NYU Shanghai High Performance Computing Administrator Zhiguo Qi. We also are grateful to OpenAI for open-sourcing their baselines codes. Chapter 4, in part, is a reprint of material from Quan Vuong, Yiming Zhang, and Keith Ross. "Supervised policy update for deep reinforcement learning". In International Conference on Learning Representation, 2019. The dissertation author is the primary author of the paper.

# Chapter 5

# Better Exploration with Optimistic Actor-Critic

## 5.1 Introduction

A major obstacle that impedes a wider adoption of actor-critic methods [LHP$^+$16, SLH$^+$14, Wil92b, SMSM99] for control tasks is their poor sample efficiency. In practice, despite impressive recent advances [HZAL18b, FHM18], millions of environment interactions are needed to obtain a reasonably performant policy for control problems with moderate complexity. In systems where obtaining samples is expensive, this often makes the deployment of these algorithms prohibitively costly.

This paper aims at mitigating this problem by more efficient exploration . We begin by examining the exploration behavior of SAC [HZAL18b] and TD3 [FHM18], two recent model-free algorithms with state-of-the-art sample efficiency and make two insights. First, in order to avoid overestimation [Has10, vHGS16], SAC and TD3 use a critic that computes an approximate lower confidence bound. The actor then adjusts the exploration policy to maximize this lower bound. This improves the stability of the updates and allows the use of larger learning rates.

However, using the lower bound can also seriously inhibit exploration if it is far from the true Q-function. If the lower bound has a spurious maximum, the covariance of the policy will decrease, causing *pessimistic underexploration*, i.e. discouraging the algorithm from sampling actions that would lead to an improvement to the flawed estimate of the critic. Moreover, Gaussian policies are *directionally uninformed*, sampling actions with equal probability in any two opposing directions from the mean. This is wasteful since some regions in the action space close to the current policy are likely to have already been explored by past policies and do not require more samples.

We formulate Optimistic Actor-Critic (OAC), an algorithm which explores more efficiently by applying the principle of optimism in the face of uncertainty [BT02]. OAC uses an off-policy exploration strategy that is adjusted to maximize an upper confidence bound to the critic, obtained from an epistemic uncertainty estimate on the Q-function computed with the bootstrap [OBPR16]. OAC avoids pessimistic underexploration because it uses an upper bound to determine exploration covariance. Because the exploration policy is not constrained to have the same mean as the target policy, OAC is directionally informed, reducing the waste arising from sampling parts of action space that have already been explored by past policies.

Off-policy Reinforcement Leaning is known to be prone to instability when combined with function approximation, a phenomenon known as the *deadly triad* [SB18, vHDS$^+$18]. OAC achieves stability by enforcing a KL constraint between the exploration policy and the target policy. Moreover, similarly to SAC and TD3, OAC mitigates overestimation by updating its target policy using a lower confidence bound of the critic [Has10, vHGS16].

Empirically, we evaluate Optimistic Actor Critic in several challenging continuous control tasks and achieve state-of-the-art sample efficiency on the Humanoid benchmark. We perform ablations and isolate the effect of bootstrapped uncertainty estimates on performance. Moreover, we perform hyperparameter ablations and demonstrate that OAC is stable in practice.

## 5.2 Preliminaries

*Reinforcement learning* (RL) aims to learn optimal behavior policies for an agent acting in an environment with a scalar reward signal. Formally, we consider a Markov decision process [Put14], defined as a tuple $(S, A, R, p, p_0, \gamma)$. An agent observes an environmental state $s \in S = \mathbb{R}^n$; takes a sequence of actions $a_1, a_2, \ldots$, where $a_t \in A \subseteq \mathbb{R}^d$; transitions to the next state $s' \sim p(\cdot|s,a)$ under the state transition distribution $p(s'|s,a)$; and receives a scalar reward $r \in \mathbb{R}$. The agent's initial state $s_0$ is distributed as $s_0 \sim p_0(\cdot)$.

A policy $\pi$ can be used to generate actions $a \sim \pi(\cdot|s)$. Using the policy to sequentially generate actions allows us to obtain a trajectory through the environment $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$. For any given policy, we define the action-value function as $Q^{\pi}(s,a) = E_{\tau : s_0 = s, a_0 = a}[\sum_t \gamma^t r_t]$, where $\gamma \in [0,1)$ is a discount factor. We assume that $Q^{\pi}(s,a)$ is differentiable with respect to the action. The objective of Reinforcement Learning is to find a deployment policy $\pi_{\text{eval}}$ which maximizes the total return $J = E_{\tau : s_0 \sim p_0}[\sum_t \gamma^t r_t]$. In order to provide regularization and aid exploration, most actor-critic algorithms [HZAL18b, FHM18, LHP+16] do not adjust $\pi_{\text{eval}}$ directly. Instead, they use a target policy $\pi_T$, trained to have high entropy in addition to maximizing the expected return $J$.[1] The deployment policy $\pi_{\text{eval}}$ is typically deterministic and set to the mean of the stochastic target policy $\pi_T$.

Actor-critic methods [SMSM99, BB00, BBW01, BB01] seek a locally optimal target policy $\pi_T$ by maintaining a *critic*, learned using a value-based method, and an *actor*, adjusted using a policy gradient update. The critic is learned with a variant of SARSA [vSvHWW09, SB18, Sut95]. In order to limit overestimation [Has10, vHGS16], modern actor-critic methods learn an approximate lower confidence bound on the Q-function [HZAL18b, FHM18], obtained by using two networks $\hat{Q}^1_{\text{LB}}$ and $\hat{Q}^2_{\text{LB}}$, which have identical structure, but are initialized with different weights. In order to avoid cumbersome terminology, we refer to $\hat{Q}_{\text{LB}}$ simply as a lower

[1]Policy improvement results can still be obtained with the entropy term present, in a certain idealized setting [HZAL18b].

bound in the remainder of the paper. Another set of target networks [MKS$^+$15b, LHP$^+$16] slowly tracks the values of $\hat{Q}_{\text{LB}}$ in order to improve stability.

$$\hat{Q}_{\text{LB}}(s_t, a_t) = \min(\hat{Q}^1_{\text{LB}}(s_t, a_t), \hat{Q}^2_{\text{LB}}(s_t, a_t)) \tag{5.1}$$

$$\hat{Q}^{\{1,2\}}_{\text{LB}}(s_t, a_t) \leftarrow R(s_t, a_t) + \gamma \min(\breve{Q}^1_{\text{LB}}(s_{t+1}, a), \breve{Q}^2_{\text{LB}}(s_{t+1}, a)) \text{ where } a \sim \pi_T(\cdot|s_{t+1}). \tag{5.2}$$

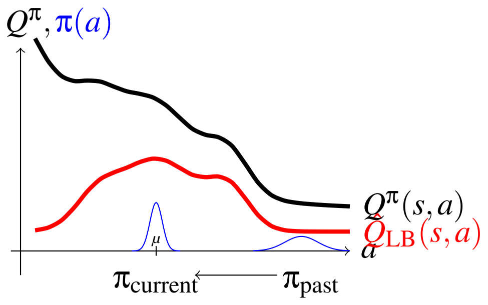
Meanwhile, the actor adjusts the policy parameter vector $\theta$ of the policy $\pi_T$ in order to maximize $J$ by following its gradient. The gradient can be written in several forms [SMSM99, SLH$^+$14, CW18, HWS$^+$15, GLT$^+$17, GLG$^+$17]. Recent actor-critic methods use a reparametrised policy gradient [HWS$^+$15, GLT$^+$17, GLG$^+$17]. We denote a random variable sampled from a standard multivariate Gaussian as $\varepsilon \sim \mathcal{N}(0, I)$ and denote the standard normal density as $\phi(\varepsilon)$. The re-parametrisation function $f$ is defined such that the probability density of the random variable $f_\theta(s, \varepsilon)$ is the same as the density of $\pi_T(a|s)$, where $\varepsilon \sim \mathcal{N}(0, I)$. The gradient of the return can then be written as:

$$\nabla_\theta J = \int_s \rho(s) \int_\varepsilon \nabla_\theta \hat{Q}_{\text{LB}}(s, f_\theta(s, \varepsilon)) \phi(\varepsilon) d\varepsilon ds \tag{5.3}$$

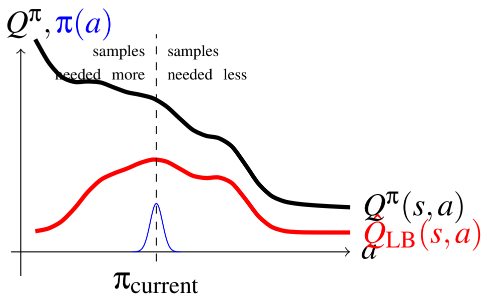
where $\rho(s) \triangleq \sum_{t=0}^{\infty} \gamma^t p(s_t = s|s_0)$ is the discounted-ergodic occupancy measure. In order to provide regularization and encourage exploration, it is common to use a gradient $\nabla_\theta J^\alpha$ that adds an additional entropy term $\nabla_\theta \mathcal{H}(\pi(\cdot, s))$.

$$\nabla_\theta J^\alpha_{\hat{Q}_{\text{LB}}} = \int_s \rho(s) \int_\varepsilon \nabla_\theta \hat{Q}_{\text{LB}}(s, f_\theta(s, \varepsilon)) \phi(\varepsilon) d\varepsilon + \alpha \underbrace{\int_\varepsilon - \nabla_\theta \log f_\theta(s, \varepsilon) \phi(\varepsilon) d\varepsilon}_{\nabla_\theta \mathcal{H}(\pi(\cdot, s))} ds \tag{5.4}$$

During training, equation 5.4 is approximated with samples by replacing integration over $\varepsilon$ with



(a) Pessimistic underexploration



(b) Directional uninformedness

**Figure 5.1**: Exploration inefficiencies in actor-critic methods. The state $s$ is fixed. The graph shows $Q^\pi$, which is unknown to the algorithm, its known lower bound $\hat{Q}_{\mathrm{LB}}$ (in red) and two policies $\pi_{\text{current}}$ and $\pi_{\text{past}}$ at different time-steps of the algorithm (in blue).

Monte-Carlo estimates and integration over the state space with a sum along the trajectory.

$$\nabla_\theta J^\alpha_{\hat{Q}_{\mathrm{LB}}} \approx \nabla_\theta \hat{J}^\alpha_{\hat{Q}_{\mathrm{LB}}} = \textstyle\sum_{t=0}^{N} \gamma^t \nabla_\theta \hat{Q}_{\mathrm{LB}}(s_t, f_\theta(s,\varepsilon_t)) + \alpha - \nabla_\theta \log f_\theta(s_t,\varepsilon_t). \tag{5.5}$$

In the standard set-up, actions used in equation 5.1 and equation 5.5 are generated using $\pi_T$. In the table-lookup case, the update can be reliably applied off-policy, using an action generated with a separate exploration policy $\pi_E$. In the function approximation setting, this leads to updates that can be biased because of the changes to $\rho(s)$. In this work, we address these issues by imposing a KL constraint between the exploration policy and the target policy. We give a more detailed account of addressing the associated stability issues in section 5.4.3.

## 5.3 Existing Exploration Strategy is Inefficient

As mentioned earlier, modern actor-critic methods such as SAC [HZAL18b] and TD3 [FHM18] explore in an inefficient way. We now give more details about the phenomena that lead to this inefficiency.

**Pessimistic underexploration.** In order to improve sample efficiency by preventing the catastrophic overestimation of the critic [Has10, vHGS16], SAC and TD3 [FHM18, HZH$^+$18, HZAL18b] use a lower bound approximation to the critic, similar to equation 5.1. However, relying on this lower bound for exploration is inefficient. By greedily maximizing the lower bound, the policy becomes very concentrated near a maximum. When the critic is inaccurate and the maximum is spurious, this can be very harmful. This is illustrated in Figure 5.1a. At first, the agent explores with a broad policy, denoted $\pi_{\text{past}}$. Since $\hat{Q}_{\text{LB}}$ increases to the left, the policy gradually moves in that direction, becoming $\pi_{\text{current}}$. Because $\hat{Q}_{\text{LB}}$ (shown in red) has a maximum at the mean $\mu$ of $\pi_{\text{current}}$, the policy $\pi_{\text{current}}$ has a small standard deviation. This is suboptimal since we need to sample actions far away from the mean to find out that the true critic $Q^{\pi}$ does not have a maximum at $\mu$.

The phenomenon of underexploration is specific to the lower as opposed to an upper bound. An upper bound which is too large in certain areas of the action space encourages the agent to explore them and correct the critic, akin to optimistic initialization in the tabular setting [SB18]. Due to overestimation, we cannot address pessimistic underexploration by simply using the upper bound in the actor [FHM18]. Instead, recent algorithms have used an entropy term equation 5.4 in the actor update. While this helps exploration somewhat by preventing the covariance from collapsing to zero, it does not address the core issue that we need to explore more around a spurious maximum. We propose a more effective solution in section 5.4.

**Directional uninformedness.** Actor-critic algorithms that use Gaussian policies, like SAC [HZH$^+$18] and TD3 [FHM18], sample actions in opposite directions from the mean with equal probability. However, in a policy gradient algorithm, the current policy will have been obtained by incremental updates, which means that it won't be very different from recent past policies. Therefore, exploration in both directions is wasteful, since the parts of the action space where past policies had high density are likely to have already been explored. This phenomenon is shown in

Figure 5.1b. Since the policy $\pi_{\text{current}}$ is Gaussian and symmetric around the mean, it is equally likely to sample actions to the left and to the right. However, while sampling to the left would be useful for learning an improved critic, sampling to the right is wasteful, since the critic estimate in that part of the action space is already good enough. In section 5.4, we address this issue by using an exploration policy shifted relative to the target policy.

## 5.4 Better Exploration with Optimism

Optimistic Actor Critic (OAC) is based on the principle of optimism in the face of uncertainty [Zie10]. Inspired by recent theoretical results about efficient exploration in model-free RL [JABJ18], OAC obtains an exploration policy $\pi_E$ which locally maximizes an approximate upper confidence bound of $Q^\pi$ each time the agent enters a new state. The policy $\pi_E$ is separate from the target policy $\pi_T$ learned using equation 5.5 and is used only to sample actions in the environment. Formally, the exploration policy $\pi_E = \mathcal{N}(\mu_E, \Sigma_E)$, is defined as

$$\mu_e, \Sigma_E = \underset{\substack{\mu,\Sigma: \\ D_{\text{KL}}((\|)N(\mu,\Sigma),\mathcal{N}(\mu_T,\Sigma_T))\leq\delta}}{\arg\max} E_{a\sim\mathcal{N}(\mu,\Sigma)}\left[\bar{Q}_{\text{UB}}(s,a)\right]. \tag{5.6}$$

Below, we derive the OAC algorithm formally. We begin by obtaining the upper bound $\bar{Q}_{\text{UB}}(s,a)$ (section 5.4.1). We then motivate the optimization problem equation 5.6, in particular the use of the KL constraint (section 5.4.2). Finally, in section 5.4.3, we describe the OAC algorithm and outline how it mitigates *pessimistic underexploration* and *directional uninformedness* while still maintaining the stability of learning. In Section 5.4.4, we compare OAC to related work.

### 5.4.1 Obtaining an Upper Bound

The approximate upper confidence bound $\bar{Q}_{\mathrm{UB}}$ used by OAC is derived in three stages. First, we obtain an epistemic uncertainty estimate $\sigma_Q$ about the true state-action value function $Q$. We then use it to define an upper bound $\hat{Q}_{\mathrm{UB}}$. Finally, we introduce its linear approximation $\bar{Q}_{\mathrm{UB}}$, which allows us to obtain a tractable algorithm.

**Epistemic uncertainty** For computational efficiency, we use a Gaussian distribution to model epistemic uncertainty. We fit mean and standard deviation based on bootstraps [ET94] of the critic. The mean belief is defined as $\mu_Q(s,a) = \frac{1}{2}\left(\hat{Q}^1_{\mathrm{LB}}(s,a) + \hat{Q}^2_{\mathrm{LB}}(s,a)\right)$, while the standard deviation is

$$\sigma_Q(s,a) = \sqrt{\sum_{i\in\{1,2\}} \tfrac{1}{2}\left(\hat{Q}^i_{\mathrm{LB}}(s,a) - \mu_Q(s,a)\right)^2} = \frac{1}{2}\left|\hat{Q}^1_{\mathrm{LB}}(s,a) - \hat{Q}^2_{\mathrm{LB}}(s,a)\right|. \tag{5.7}$$

The bootstraps are obtained using equation 5.1. Since existing algorithms [HZAL18b, FHM18] already maintain two bootstraps, we can obtain $\mu_Q$ and $\sigma_Q$ at negligible computational cost. Despite the fact that equation 5.1 uses the same target value for both bootstraps, we demonstrate in Section 5.5 that using a two-network bootstrap leads to a large performance improvement in practice. Moreover, OAC can be easily extended to to use more expensive and better uncertainty estimates if required.

**Upper bound.** Using the uncertainty estimate equation 5.7, we define the upper bound as $\hat{Q}_{\mathrm{UB}}(s,a) = \mu_Q(s,a) + \beta_{\mathrm{UB}}\sigma_Q(s,a)$. We use the parameter $\beta_{\mathrm{UB}} \in \mathbb{R}^+$ to fix the level of optimism. In order to obtain a tractable algorithm, we approximate $\hat{Q}_{\mathrm{UB}}$ with a linear function $\bar{Q}_{\mathrm{UB}}$.

$$\bar{Q}_{\mathrm{UB}}(s,a) = a^\top \left[\nabla_a \hat{Q}_{\mathrm{UB}}(s,a)\right]_{a=\mu_T} + \mathrm{const} \tag{5.8}$$

By Taylor's theorem, $\bar{Q}_{\text{UB}}(s,a)$ is the best possible linear fit to $\hat{Q}_{\text{UB}}(s,a)$ in a sufficiently small region near the current policy mean $\mu_T$ for any fixed state $s$ [Cal10, Theorem 3.22]. Since the gradient $\left[\nabla_a \hat{Q}_{\text{UB}}(s,a)\right]_{a=\mu_T}$ is computationally similar to the lower-bound gradients in equation 5.5, our upper bound estimate can be easily obtained in practice without additional tuning.

### 5.4.2 Optimistic Exploration

Our exploration policy $\pi_E$, introduced in equation 5.6, trades off between two criteria: the maximization of an upper bound $\bar{Q}_{\text{UB}}(s,a)$, defined in equation 5.8, which increases our chances of executing informative actions, according to the principle of *optimism in the face of uncertainty* [BT02], and constraining the maximum KL divergence between the exploration policy and the target policy $\pi_T$, which ensures the stability of updates. The KL constraint in equation 5.6 is crucial for two reasons. First, it guarantees that the exploration policy $\pi_E$ is not very different from the target policy $\pi_T$. This allows us to preserve the stability of optimization and makes it less likely that we take catastrophically bad actions, ending the episode and preventing further learning. Second, it makes sure that the exploration policy remains within the action range where the approximate upper bound $\bar{Q}_{\text{UB}}$ is accurate. We chose the KL divergence over other similarity measures for probability distributions since it leads to tractable updates.

Thanks to the linear form on $\bar{Q}_{\text{UB}}$ and because both $\pi_E$ and $\pi_T$ are Gaussian, the maximization of equation 5.6 can be solved in closed form. We state the solution below.

**Proposition 1.** *The exploration policy resulting from equation 5.6 has the form $\pi_E = \mathcal{N}(\mu_E, \Sigma_E)$, where*

$$\mu_E = \mu_T + \frac{\sqrt{2\delta}}{\left\| \left[\nabla_a \hat{Q}_{UB}(s,a)\right]_{a=\mu_T} \right\|_{\Sigma}} \Sigma_T \left[\nabla_a \hat{Q}_{UB}(s,a)\right]_{a=\mu_T} \quad \textit{and} \quad \Sigma_E = \Sigma_T. \tag{5.9}$$

We stress that the covariance of the exploration policy is the same as the target policy.

**Algorithm 4** Optimistic Actor-Critic (OAC).

```
Require: $w_1$, $w_2$, $\theta$          ▷ Initial parameters $w_1, w_2$ of the critic and $\theta$ of the target policy $\pi_T$.
 1: $\breve{w}_1 \leftarrow w_1, \breve{w}_2 \leftarrow w_2, \mathcal{D} \leftarrow \emptyset$          ▷ Initialize target network weights and replay pool
 2: for each iteration do
 3:     for each environment step do
 4:         $a_t \sim \pi_E(a_t|s_t)$          ▷ Sample action from exploration policy as in equation 5.9.
 5:         $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$          ▷ Sample transition from the environment
 6:         $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, R(s_t, a_t), s_{t+1})\}$          ▷ Store the transition in the replay pool
 7:     end for
 8:     for each training step do
 9:         for $i \in \{1, 2\}$ do          ▷ Update two bootstraps of the critic
10:             update $w_i$ with $\hat{\nabla}_{w_i} \|\hat{Q}^i_{\text{LB}}(s_t,a_t) - R(s_t,a_t) - \gamma \min(\breve{Q}^1_{\text{LB}}(s_{t+1},a), \breve{Q}^2_{\text{LB}}(s_{t+1},a))\|^2_2$
11:         end for
12:         update $\theta$ with $\nabla_\theta \hat{J}^{\alpha}_{\hat{Q}'_{\text{LB}}}$          ▷ Policy gradient update.
13:         $\breve{w}_1 \leftarrow \tau w_1 + (1-\tau)\breve{w}_1, \breve{w}_2 \leftarrow \tau w_2 + (1-\tau)\breve{w}_2$          ▷ Update target networks
14:     end for
15: end for
Output: $w_1$, $w_2$, $\theta$          ▷ Optimized parameters
```

### 5.4.3 The Optimistic Actor-Critic Algorithm

Optimistic Actor Critic (see Algorithm 4) samples actions using the exploration policy equation 5.9 in line 4 and stores it in a memory buffer. The term $\left[\nabla_a \hat{Q}_{\text{UB}}(s,a)\right]_{a=\mu_T}$ in equation 5.9 is computed at minimal cost[2] using automatic differentiation, analogous to the critic derivative in the actor update equation 5.4. OAC then uses its memory buffer to train the critic (line 10) and the actor (line 12). We also introduced a modification of the lower bound used in the actor, using $\hat{Q}'_{\text{LB}} = \mu_Q(s,a) + \beta_{\text{LB}}\sigma_Q(s,a)$, allowing us to use more conservative policy updates. The critic equation 5.1 is recovered by setting $\beta_{\text{LB}} = -1$.

**OAC avoids the pitfalls of greedy exploration** Figure 5.2 illustrates OAC's exploration policy $\pi_E$ visually. Since the policy $\pi_E$ is far from the spurious maximum of $\hat{Q}_{\text{LB}}$ (red line in figure 5.2), executing actions sampled from $\pi_E$ leads to a quick correction to the critic estimate. This way, *OAC avoids pessimistic underexploration*. Since $\pi_E$ is not symmetric with respect to the mean of

[2]In practice, the per-iteration wall clock time it takes to run OAC is the same as SAC.

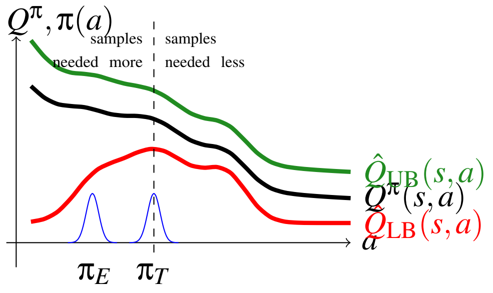


**Figure 5.2**: The OAC exploration policy $\pi_E$ avoids pessimistic underexploration by sampling far from the spurious maximum of the lower bound $\hat{Q}_{\text{LB}}$. Since $\pi_E$ is not symmetric wrt. the mean of the target policy (dashed line), it also addresses directional uninformedness.

$\pi_T$ (dashed line), *OAC also avoids directional uninformedness*.

**Stability** While off-policy deep Reinforcement Learning is difficult to stabilize in general [SB18, vHDS$^+$18], OAC is remarkably stable. Due to the KL constraint in equation equation 5.6, the exploration policy $\pi_E$ remains close to the target policy $\pi_T$. In fact, despite using a separate exploration policy, OAC isn't very different in this respect from SAC [HZAL18b] or TD3 [FHM18], which explore with a stochastic policy but use a deterministic policy for evaluation. In Section 5.5, we demonstrate empirically that OAC and SAC are equally stable in practice. Moreover, similarly to other recent state-of-the-art actor-critic algorithms [FHM18, HZH$^+$18], we use target networks [MKS$^+$15b, LHP$^+$16] to stabilize learning.

**Overestimation vs Optimism** While OAC is an optimistic algorithm, it does not exhibit catastrophic overestimation [FHM18, Has10, vHGS16]. OAC uses the optimistic estimate equation 5.8 for exploration only. The policy $\pi_E$ is computed from scratch (line 4 in Algorithm 4) every time the algorithm takes an action and is used only for exploration. The critic and actor updates equation 5.1 and equation 5.5 are still performed with a lower bound. This means that there is no way the upper bound can influence the critic except indirectly through the distribution of state-action pairs in the memory buffer.

### 5.4.4 Related work

OAC is distinct from other methods that maintain uncertainty estimates over the state-action value function. Actor-Expert [LJL$^+$18] uses a point estimate of $Q^\star$, unlike OAC, which uses a bootstrap approximating $Q^\pi$. Bayesian actor-critic methods [GE07, GMPT15, GEV16] model the probability distribution over $Q^\pi$, but unlike OAC, do not use it for exploration. Approaches combining DQN with bootstrap [CSAS17, OAC18] and the uncertainty Bellman equation [OOMM18] are designed for discrete actions. Model-based reinforcement learning methods thet involve uncertainty [GMR16, DHLDVU16, CCML18] are very computationally expensive due to the need of learning a distribution over environment models. OAC may seem superficially similar to natural actor critic [Ama98, Kak01, PS06, PS08b] due to the KL constraint in equation 5.6. In fact, it is very different. While natural actor critic uses KL to enforce the similarity between infinitesimally small updates to the target policy, OAC constrains the *exploration policy* to be within a non-trivial distance of the target policy. Other approaches that define the exploration policy as a solution to a KL-constrained optimization problem include MOTO [ALP$^+$15], MORE [ANAA16] and Maximum a Posteriori Policy optimization [AST$^+$18b]. These methods differ from OAC in that they do not use epistemic uncertainty estimates and explore by enforcing entropy.

## 5.5 Experiments

Our experiments have three main goals. First, to test whether Optimistic Actor Critic has performance competitive to state-of-the art algorithms. Second, to assess whether optimistic exploration based on the bootstrapped uncertainty estimate equation 5.7, is sufficient to produce a performance improvement. Third, to assess whether optimistic exploration adversely affects the stability of the learning process.

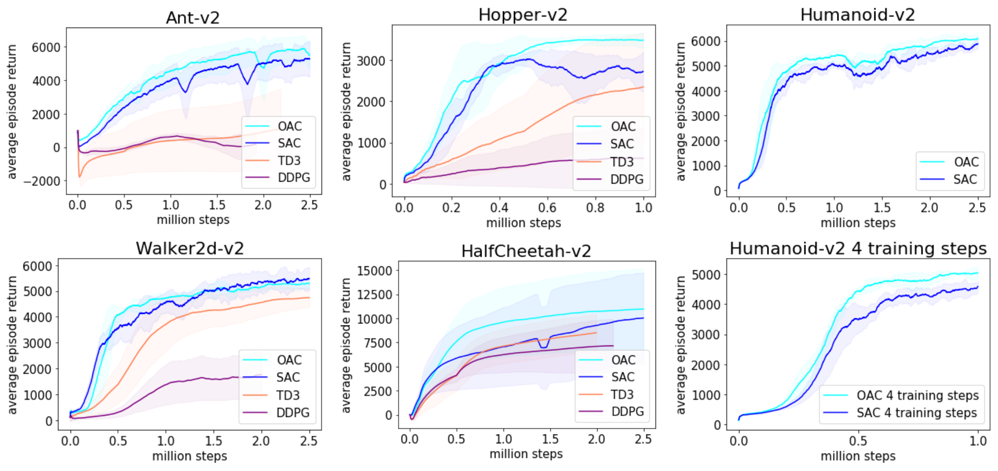


**Figure 5.3**: OAC versus SAC, TD3, DDPG on 5 Mujoco environments. The horizontal axis indicates number of environment steps. The vertical axis indicates the total undiscounted return. The shaded areas denote one standard deviation.

**MuJoCo Continuous Control** We test OAC on the MuJoCo [TET12] continuous control benchmarks. We compare OAC to SAC [HZH$^+$18] and TD3 [FHM18], two recent model-free RL methods that achieve state-of-the art performance. For completeness, we also include a comparison to a tuned version of DDPG [LHP$^+$16], an established algorithm that does not maintain multiple bootstraps of the critic network. OAC uses 3 hyper-parameters related to exploration. The parameters $\beta_{\mathrm{UB}}$ and $\beta_{\mathrm{LB}}$ control the amount of uncertainty used to compute the upper and lower bound respectively. The parameter $\delta$ controls the maximal allowed divergence between the exploration policy and the target policy. Results in Figure 5.3 show that using optimism improves the overall performance of actor-critic methods. On Ant, OAC improves the performance somewhat. On Hopper, OAC achieves state-of the art final performance. On Walker, we achieve the same performance as SAC while the high variance of results on HalfCheetah makes it difficult to draw conclusions on which algorithm performs better.

**State-of-the art result on Humanoid** The upper-right plot of Figure 5.3 shows that the vanilla version of OAC outperforms SAC the on the Humanoid task. To test the statistical significance of our result, we re-ran both SAC and OAC in a setting where 4 training steps per iteration are used.

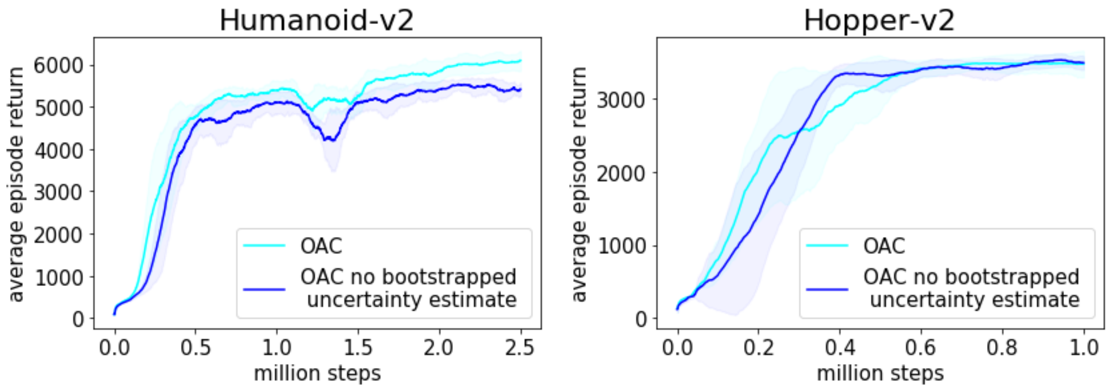


**Figure 5.4**: Impact of the bootstrapped uncertainty estimate on the performance of OAC.

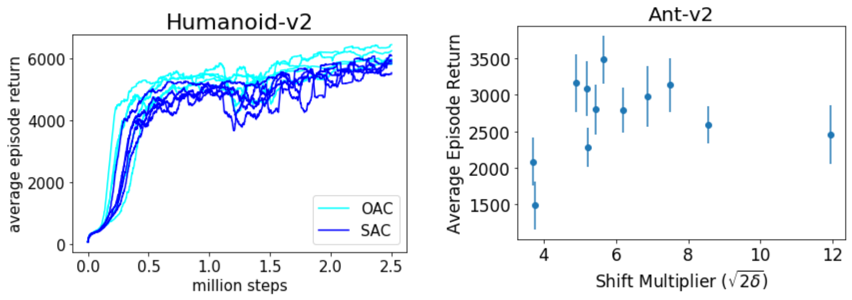


**Figure 5.5**: Left figure: individual runs of OAC vs SAC. Right figure: sensitivity to the KL constraint $\delta$. Error bars indicate 90% confidence interval.

By exploiting the memory buffer more fully, the 4-step versions show the benefit of improved exploration more clearly. The results are shown in the lower-right plot in Figure 5.3. At the end of training, the 90% confidence interval for the performance of OAC was $5033 \pm 147$ while the performance of SAC was $4586 \pm 117$. We stress that we did not tune hyper-parameters on the Humanoid environment. Overall, the fact that we are able to improve on Soft-Actor-Critic, which is currently the most sample-efficient model-free RL algorithm for continuous tasks shows that optimism can be leveraged to benefit sample efficiency.

**Usefulness of the Bootstrapped Uncertainty Estimate** OAC uses an epistemic uncertainty estimate obtained using two bootstraps of the critic network. To investigate its benefit, we compare the performance of OAC to a modified version of the algorithm, which adjusts the exploration policy to maximize the approximate lower bound, replacing $\hat{Q}_{\text{UB}}$ with $\hat{Q}_{\text{LB}}$ in equation

equation 5.9. While the modified algorithm does not use the uncertainty estimate, it still uses a shifted exploration policy, preferring actions that achieve higher state-action values. The results is shown in Figure 5.4. Using the bootstrapped uncertainty estimate improves performance on the most challenging Humanoid domain, while producing either a slight improvement or a no change in performance on others domains. Since the upper bound is computationally very cheap to obtain, we conclude that it is worthwhile to use it.

**Sensitivity to the KL constraint** OAC relies on the hyperparameter $\delta$, which controls the maximum allowed KL divergence between the exploration policy and the target policy. In Figure 5.5, we evaluate how the term $\sqrt{2\delta}$ used in the the exploration policy equation 5.9 affects average performance of OAC trained for 1 million environment steps on the Ant-v2 domain. The results demonstrate that there is a broad range of settings for the hyperparameter $\delta$, which leads to good performance.

**Learning is Stable in Practice** Since OAC explores with a shifted policy, it might at first be expected of having poorer learning stability relative to algorithms that use the target policy for exploration. While we have already shown above that the performance difference between OAC and SAC is statistically significant and not due to increased variance across runs, we now investigate stability further. In Figure 5.5 we compare individual learning runs across both algorithms. We conclude that OAC and SAC are similarly stable, avoiding the problems associated with stabilising deep off-policy RL [SB18, vHDS$^+$18].

## 5.6 Conclusions

We present Optimistic Actor Critic (OAC), a model-free deep reinforcement learning algorithm which explores by maximizing an approximate confidence bound on the state-action value function. By addressing the inefficiencies of *pessimistic underexploration* and *directional* *uninformedness*, we are able to achieve state-of-the art sample efficiency in continuous control tasks. Our results suggest that the principle of optimism in the face of uncertainty can be used to improve the sample efficiency of policy gradient algorithms in a way which carries almost no additional computational overhead.

## 5.7 Acknowledgement

The work presented in this Chapter was carried out while the dissertation author was a research intern at Microsoft Research Cambridge, UK working under the supervision of Dr. Kamil Ciosek. Chapter 5, in part, is a reprint of material from Kamil Ciosek, Quan Vuong, Robert Loftin and Katja Hofmann. "Better Exploration with Optimistic Actor-Critic". In Conference on Neural Information Processing Systems, 2019.

# Chapter 6

# Single RGB-D Camera Teleoperation for General Robotic Manipulation

## 6.1 Introduction

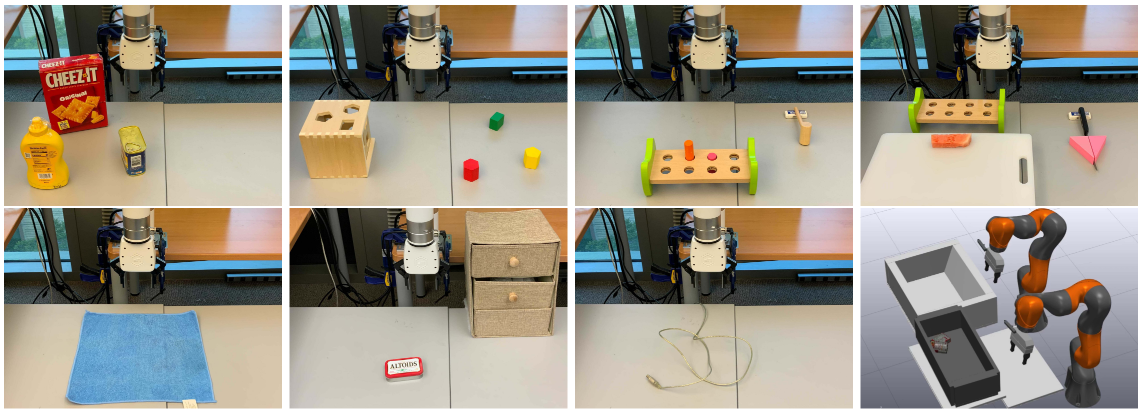


**Figure 6.1**: We propose a teleoperation system which uses a single RGB-D camera to capture human intent. We also show successful pick-and-place trials while only using RGB without depth sensing. The human operator does not have access to any form of tactile feedback. Tasks (a)-(g) are evaluated on a real robot arm. Dual arm task (h) is performed in simulation.

Teleoperation is an essential interaction mode for many robot systems. As we remove fences and allow robots to operate in mixed environments, it is a challenge to provide fully autonomous operation. We are increasing seeing systems where ”sometimes” the last inch of motion is a challenge to perform robustly. For level 4 cars, the expectation is that operators will take over control as needed. For manipulation tasks, the same might be true. With ML based programming, pure ”learning by demonstration” might not always be possible due to safety consideration, access to workspace, etc. In many use-cases, teleoperation of a robot is a convenient modality for interaction with the robot.

Design of a system for teleoperation for remote-operation, intervention or task learning all have to consider three aspects: i) Detection of operator actions to control the robot, interpretation of the operator intent, ii) mapping of intent to task coordinates, and iii) coordinated execution of operator commands and providing feedback.

Recently, cameras have become omnipresent due to low price and access to computing power to process data. To capture the intents of the users, we propose a minimalist approach which uses single RGB-D camera to capture the intents expressed as changes in users hand poses. Frequently, the operator and the robot do not operate in the same coordinate frames. Using oblique coordinates is a flexible and convenient mechanism to define a mapping from operator to robot frames. Defining the operator reference frame using oblique coordinates also allows for motion repeatability between the reference frame definition procedure prior to teleoperation and during teleoperation trials. Finally, for execution of commands, there is a need to make part of the control local. Physical interaction is close to impossible to perform in a closed loop involving the operator, especially when the system has any level of delays.

We present a methodology to teleoperate a physical robot manipulator. We discuss two approaches to hand detection, tracking and intent recognition. We transfer commands from user coordinates to task space using oblique coordinates. Finally, we describe the design of a control system for physical task execution. We demonstrate the performance of the system using both simulations and real-world tests.

A key assumption of our approach and also of prior works [MSPK13, VJK12, SGJ$^+$12, DZML12] is the availability of the depth modality in the camera observation to estimate the 3D positions of skeletal keypoints. By approximating the true camera projection model with a weak perspective transformation, our system enables the operator to perform pick-and-place tasks using only RGB and without depth sensing. Differently from previous works which use the Kinect and the accompanying proprietary human pose tracker for tele-operation, we use state-of-the-art and open-sourced human pose estimation machine learning models to estimate the hand poses.

Recordings of teleoperation trials are available at https://sites.google.com/view/manipulation-teleop-with-rgbd

## 6.2 Related Work

In the 50s, Raymond Goertz developed teleoperation systems of pairs of mechanically linked parent-children robots. These systems allowed operators to handle radioactive material and transmitted forces through the connected mechanical system. Such connectivity limited the physical distance between the operator and the robot. Teleoperation systems have found widespread applications [SK07].

To extend the operators-robots physical distance and reduce the needs for special purpose hardware to capture the operators' intents, researchers turned their attention to vision-based motion capture system. Pioneering motion capture works use special-purpose markers or gloves to track the 2D or 3D positions of critical points [Dor94, FSL$^+$17, TAH$^+$04]. The tracking results can be further combined with Virtual Reality devices for better feedback to human operators [WP09]. The marker-based solution often suffers from self-occlusion, thus redundant markers are often placed on the gloves to improve tracking consistency. Additional hardware, such as inertia measurement unit [MJKM04], electromyography [LLLY19], and haptic devices [KT15], can be used to further improve tracking performance. Tactile feedback [BRS04, JF09] to the human operator provided by specially-design gloves also improves the controllability of the teleoperated robots. While these systems offer good teleoperation performance, they require the human operator to have access to hardware that are not widely available, and are thus difficult to deploy widely.

Due to their less stringent hardware requirement, vision-based markerless hand tracking has received significant attention from the teleoperation community. [SKR+15, LML+19, AGHK18] tracks the hand poses via monocular camera or depth sensor. A hand model can be also be used to improve tracking performance [KVW07, DZML12]. These works use the hand pose tracking results to control a robot arm with parallel gripper for simple tasks, e.g. pick and place. In contrast, our teleoperation system can perform complex tasks such as cutting and cloth folding. DexPilot [HVWY+20] also demonstrates successful completion of complex tasks using a multi-finger hand. However, DexPilot requires high tracking accuracy of most finger joints. The human operators require access to four RGB-D cameras with calibrated extrinsic to register the complete points of human hand. The human operator in our teleoperation system can perform complex tasks while using only 1 RGB-D camera without extrinsic calibration. We however recognize that the multi-fingered setting studied in DexPilot is significantly more challenging than our parallel yaw setting.

Both marker-based and vision-based teleoperation system make use of hand pose estimation techniques, which have advanced rapidly due to the integration of deep learning and hand models. Motivated by the success of blend skinning techniques [LMR+15, KŽ05], MANO [RTB17] models shape variation from hand scans by learning pose dependent blend shapes and represents the hand as a combination of shape parameters and pose parameters. [BBT19, LJX+21, RSJ20] also adopt linear blend skinning model for hand pose estimation and tracking. Our teleoperation system illustrates that state-of-the-art vision-based hand pose estimation techniques are accurate enough to allow for the completion of tasks that require mm-level precision, such as peg-in-hole.

Given the estimated human hand poses, a wide spectrum of possible motion correspondence strategy exists to compute the desired robot poses. Our work uses direct motion control

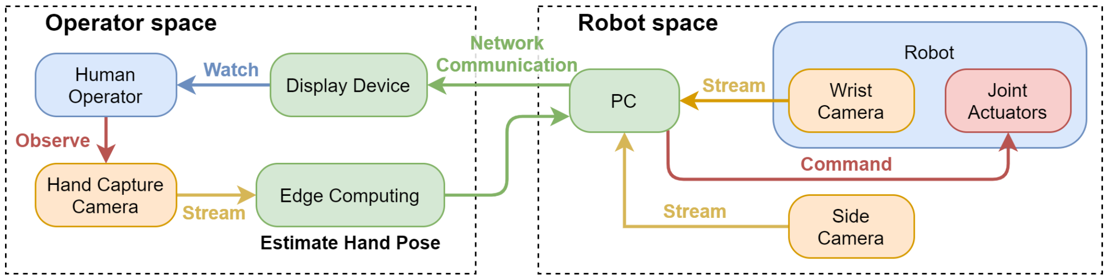


**Figure 6.2**: Control flow of our teleoperation system. The wrist and side camera in the robot workspace send RGB streams to a display device in the operator space. The operator watches the display device and move their hands to signal intent to the robot controller. The hand capture camera in the operator space observes the hands movement and sends the RGB-D streams to an edge computing device. The device then estimates the hand poses, transforms the changes in human hand poses to changes in desired robot poses, and send the desired changes to the computer in the robot space controlling the robot motion.

[SK07]. Another common strategy is gesture-based control, where the operator communicates their intent through changes in hand gestures, instead of changes in hand poses. In such system, the changes in robot motion is often a pre-defined and fixed function of the changes in hand gestures. As such, the teleoperation system can limit the operators to perform simple task such as grasping [OTM21] or limit the degree of freedom of movement of the end effector [JKP21]. Our work demonstrates successful completion of complex tasks such as cutting and also allows for controlling the 6 degrees of freedom of the end effector. A gesture-based teleoperation system also requires the operator to remember the mapping between hand gestures and robot motion [ZTC$^+$20].

## 6.3 Architecture

Figure 6.2 illustrates the control flow of our tele-operation system. In this section, we explain the hand pose estimation models, the mapping from operator workspace to robot workspace (which we refer to as *motion correspondence*), dynamic motion scaling, and the hardware system.

### 6.3.1 Hand pose estimation from RGB-D observation

The hand pose estimator takes as input a RGB-D frame and outputs 21 2D positions of hand keypoints in pixel space and the wrist pose in camera frame for each hand. Compared to previous works which use the built-in pose tracker in the Kinect sensor [VJK12, SGJ$^+$12, DZML12], we use state-of-the-art machine learning models to estimate the hand poses. Such approach allows us to implement our system on commodity camera, is not restricted to the Kinect sensor and thus might be more scalable. Given the RGB frame, we use MediaPipe [ZBV$^+$20] to estimate the 2D positions of hand keypoints. Given the keypoints, we compute axis-aligned bounding-box to crop the image around hand region. The cropped images are then input into the Frankmocap [RSJ20] hand pose regressor to predict the orientation of the wrist in camera frame. We also obtain the 3D positions of the wrist by un-projecting the coordinates of the corresponding keypoint using the depth map. The hand pose estimator is implemented as a ROS [QCG$^+$09] node with publish rate of $10Hz$. Given the 2D positions in pixel space of the hand keypoints, we send a command to the robot controller to close the robot parallel yaw gripper if the distance between the right thumb and index tips falls below a pre-defined threshold [DZML12, KVW07].

### 6.3.2 Hand pose estimation from RGB without depth sensing

A key limitation of Kinect-based approach [VJK12, SGJ$^+$12, DZML12] is the requirement of depth sensing to estimate the 3D positions of skeletal keypoints. Using RGB instead of RGB-D camera is a promising extension to our method. The main challenge is to obtain accurate 3D positions of operator wrists without depth information. Monocular depth estimation is in general a challenging problem. However, given that we only requires the 3D positions of the wrists, we were nevertheless able to obtain promising results. In addition, the neural network based hand pose estimation models have strong prior over the shape and size of the human hand.

To estimate 3D hand pose from RGB, we crop the image around the hand region and

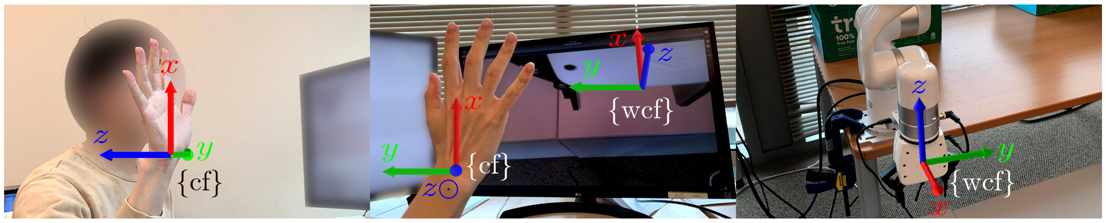


**Figure 6.3**: Illustration of the Cartesian operator frame and the robot wrist camera frame. The left and middle figures illustrate the axes of the Cartesian operator frame $\{cf\}$ as seen from the operator space camera and by the operator respectively. The middle and right figures illustrate the robot wrist camera frame $\{wcf\}$ as seen by the operator and in the robot space. The alignment between the operator frame $\{cf\}$ and camera frame $\{wcf\}$ from the perspective of the operator, as shown by the middle figure, mitigates viewpoint and reference frame mismatch between the operator and the robot.

predict the weak perspective transformation scale $s_h$ using the model from FrankMocap [RSJ20]. The weak perspective transformation approximates the true camera projection model. It assumes that for a small object, such as human hand, the depth value of any point on this object is the same. If the assumption holds, the pixel coordinates from weak projection well approximate the real projection model. We can thus approximate the depth simply by $\frac{Cf}{s_h}$, where $f$ is the focal length, $C$ is a scalar constant. Given the estimated depth and camera projection model, we can obtain the 3D position of the wrist expressed in the operator workspace camera frame.

### 6.3.3 Motion correspondence

The motion correspondence module computes the desired pose of the robot end effector given the current estimates of the operator hand poses. To do so, the motion correspondence module uses a define-at-runtime reference frame in the operator workspace. In this sub-section, we motivate the need for the reference frame, introduce its Cartesian and non-Cartesian oblique instantiations and describe how they are used to find the desired robot end effector pose. We refer to the reference frame in the operator workspace as the operator frame. We will use monogram notation [TtDDT] in the subsequent description of the motion correspondence module.

**Why is the operator frame useful?** An issue that frequently appears in teleoperation is viewpoint mismatch. Human operators naturally interpret their motion commands to the robots with respect to an egocentric frame. If the axes of the egocentric frame are not aligned with the axes of the frame in the robot workspace used to represent the desired robot end effector pose, the operators need to understand the relationship between their egocentric frames and the robots' frame of reference and perform mental rotation during teleoperation [KH01, SWD10, MCM$^+$17]. Such calculation severely impedes the ease of teleoperation. To mitigate the issue, our system allows the operator to define an egocentric frame that aligns with the frame in the robot workspace used for representing the desired robot end effector pose. We refer to such an egocentric frame in the operator workspace as an operator frame.

**Constructing the Cartesian operator frame.** In our system, the desired robot end effector pose is represented with respect to a robot wrist camera frame $\{wcf\}$, illustrated in Figure 6.3. Our system defines the wrist camera frame $\{wcf\}$ prior to teleoperation and keeps it fixed. Given the robot wrist camera frame $\{wcf\}$, our system defines the Cartesian operator frame in the operator workspace as follows. For each axis of the robot wrist camera frame $\{wcf\}$, the system asks the operator through a GUI to move their right wrist in a direction such that if the operator moves their wrist in the same direction during teleoperation, the end effector will move in the direction of the corresponding axis of the robot wrist camera frame $\{wcf\}$. As such, the frame with which the desired robot end effector pose is represented with respect to is aligned with the egocentric Cartesian operator frame, thus mitigating the viewpoint mismatch issue.

More concretely, let $\{c\}$ represents the frame of the camera in the operator workspace. During the Cartesian operator frame definition procedure, the hand pose estimator predicts the positions of the operator right wrist with respect to the operator workspace camera frame $\{c\}$. Our system thus obtains three sets of wrist positions, one for each axis of the robot wrist camera frame $\{wcf\}$. For each set of positions, we perform RANSAC line fitting to each set and then normalize the resulting lines to unit length. Let the resulting unit vector corresponding to the $x$-axis of the wrist camera frame $\{wcf\}$ be $n_x \in R^3$. We similarly estimate $n_y, n_z \in R^3$. We then define the matrix $\left[n_x, n_y, n_z\right] \in R^{3\times 3}$ with the column vectors being $n_x, n_y, n_z$ and project the matrix to the closet orthogonal matrix $R \in SO(3)$. We also compute the point $p$ closest to the three vectors $n_x, n_y, n_z$ in a least square sense. Let $\{cf\}$ denotes the Cartesian operator frame. The procedure above produces $p$ and $R$, which together form the pose ${}^{c}T^{cf} = (p, R) \in SE(3)$ of the Cartesian operator frame $\{cf\}$ with respect to the operator workspace camera frame $\{c\}$.

**Motion correspondence between operator and robot.** Given the pose of the Cartesian operator frame ${}^{c}T^{cf}$, we now describe how to compute the desired pose of the end effector given the current pose of the operator right hand wrist during teleoperation. Let $\{wr\}$ denotes the frame attached to the operator right hand wrist. Let $\{w\}, \{e\}$ denote the robot world and end effector frames respectively. Given ${}^{c}T^{wr}$, we need to compute ${}^{w}T^{e}_{desired}$. We compute the desired position and orientation separately. Given the current position of the right hand wrist ${}^{c}p^{wr}$, we first express it with respect to the Cartesian operator frame: ${}^{cf}p^{wr} = {}^{cf}T^{c} \cdot {}^{c}p^{wr}$.

We then treat a scaled version of ${}^{cf}p^{wr}$ to be the desired end effector position with respect to the robot wrist camera frame $\{wcf\}$: ${}^{wcf}p^{e} = \alpha \cdot {}^{cf}p^{wr}$.

Since our system defines the robot wrist camera frame $\{wcf\}$ prior to teleoperation and keeps the pose of $\{wcf\}$ with respect to the robot world frame $\{w\}$ fixed during teleoperation, the desired end effector position is:

$${}^{w}p^{e}_{desired} = {}^{w}T^{wcf} \cdot {}^{wcf}p^{e}$$

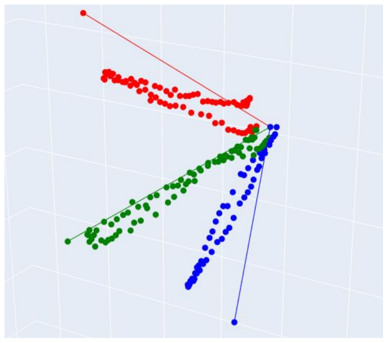
$\alpha$ is a scalar to control the scaling between human and robot motion. If $\alpha = 1$, the end effector moves the same distance as the operator wrist. The appropriate value of $\alpha$ depends on the task. For task like peg in hole, we might want to use a smaller $\alpha$ to allow for more precise robot control.

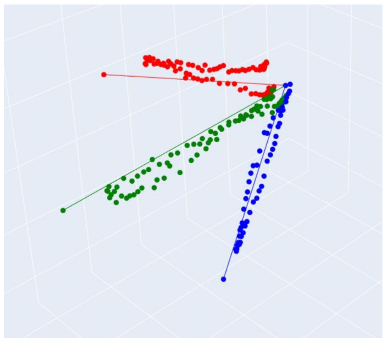
Similarly, given the orientation ${}^{c}R^{wr}$ of the operator right hand wrist frame $\{wr\}$ with respect to the operator space camera frame $\{c\}$, we represent the orientation ${}^{c}R^{wr}$ with respect to the Cartesian operator frame $\{cf\}$. We then use the wrist camera frame $\{wcf\}$ to find the desired orientation of the end effector in the robot workspace world frame ${}^{w}R^{e}_{desired}$.

**Oblique operator frame.** To construct the Cartesian operator frame $\{cf\}$, for each axis of the robot wrist camera frame $\{wcf\}$, our system asks the operator through a GUI to move their right wrist in a direction such that if the operator moves their wrist in the same direction during teleoperation, the end effector will move along the same axis of the wrist camera frame $\{wcf\}$. However, such promise by the system is often not possible because the 3 best fit vectors $n_x, n_y, n_z$ to the operator wrist positions are usually not orthogonal. This necessitates the projection step to $SO3$ as discussed above. However, projecting to $SO3$ the matrix whose columns are the three best-fit vectors $n_x, n_y, n_z$ implies that as the operator moves their wrist in the same direction as one of the best fit vector, the robot end effector might not move along the corresponding axes of the wrist camera frame $\{wcf\}$. In other words, if the operator repeats the same motion during the operator frame definition procedure and during teleoperation, the robot end effector might move in different directions. We therefore introduce the use of non-Cartesian oblique coordinate frame to ensure motion repeatability.

An oblique frame is a frame whose axes are not orthogonal [Fro]. The difference between Cartesian and oblique coordinate frames are further illustrated in Figure 6.4. Given an arbitrary point in space, to find the measure number of a point with respect to an axis of an oblique coordinate frame, we project the point onto the axis along the direction parallel to the hyper-plane defined by the remaining axes.

Let $\{of\}$ denotes the oblique operator frame. We next explain how to obtain the oblique frame $\{of\}$ from the cartesian frame $\{cf\}$. We express the axes of the oblique coordinate frame as vector in the Cartesian operator frame. To find the x-axis of the oblique coordinate frame, we find the vector passing through the origin of the Cartesian operator frame and is parallel to the best fit vector $n_x$ of the 3d wrist positions obtained when defining the x-axis of the Cartesian operator frame. Denote the x-axis of the oblique frame in the Cartesian frame by ${}^{cf}of_x \in R^3$. Similarly, we can obtain the y-axis and z-axis of the oblique frame ${}^{cf}of_y, {}^{cf}of_z \in R^3$. The normal


(a) Cartesian operator frame. The z-axis of the Cartesian operator frame (blue vector) is not parallel with the best fit lines of the blue points. Let the blue points represent the captured operator wrist positions during the definition procedure of the z-axis of the Cartesian operator frame. If the operator moves their wrist again along the direction represented by the blue points during teleoperation, the robot end effector would move in the direction represented by the blue vector. This is an issue because of the lack of motion repeatability between operator frame definition and during teleoperation trials. We discuss this issue in more details in subsection 6.3.3 under **Oblique operator frame**.


(b) Oblique operator frame. The axes of the oblique operator frame, indicated by the 3 vectors, are the best fit vectors to the three sets of captured wrist positions during the operator frame definition procedure.

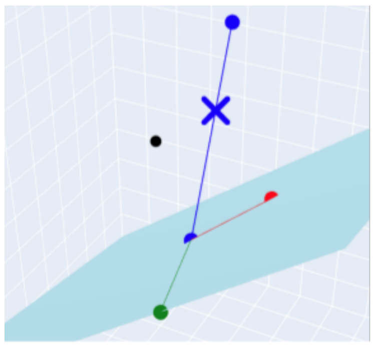

(c) Computing coordinates in oblique frame: the black point is projected onto the z-axis, indicated by blue cross, along the direction parallel to the hyperplane defined by the x and y axes.

**Figure 6.4**: An illustration of the the differences between Cartesian and oblique operator frame.

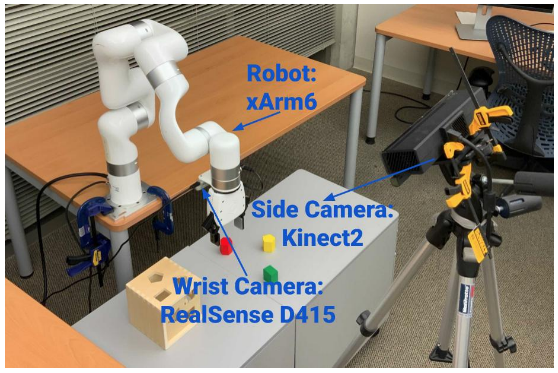
to the hyperplane defined by the x and y axes of the oblique frame is ${}^{cf}n_{xy} ={}^{cf} of_x \times {}^{cf} of_y$. Similarly, the normal the hyperplane defined by the remaining pairs of axes are ${}^{cf}n_{xz}$ and ${}^{cf}n_{yz}$. Having defined these quantities, the computation to transform the position ${}^{c}p^{wr}$ of operator wrist represented with respect to the camera frame in operator workspace to desired robot end effector position in the robot workspace world frame ${}^{w}p^{e}_{desired}$ is:

$$
\begin{aligned}
&{}^{cf}p^{wr} ={}^{cf} T^{c}.{}^{c}p^{wr}\\
&{}^{of}p^{wr}_{x} = \left\langle {}^{cf}p^{wr}, {}^{cf}n_{yz} \right\rangle \Big/ \left\langle {}^{cf}of_{x}, {}^{cf}n_{yz} \right\rangle\\
&{}^{of}p^{wr}_{y} = \left\langle {}^{cf}p^{wr}, {}^{cf}n_{xz} \right\rangle \Big/ \left\langle {}^{cf}of_{y}, {}^{cf}n_{xz} \right\rangle\\
&{}^{of}p^{wr}_{z} = \left\langle {}^{cf}p^{wr}, {}^{cf}n_{xy} \right\rangle \Big/ \left\langle {}^{cf}of_{z}, {}^{cf}n_{xy} \right\rangle\\
&{}^{wcf}p^{e} = \alpha. \left[ {}^{of}p^{wr}_{x}, {}^{of}p^{wr}_{y}, {}^{of}p^{wr}_{z} \right] \in R^{3\times 1}\\
&{}^{w}p^{e}_{desired} ={}^{w} T^{wcf}.{}^{wcf}p^{e}
\end{aligned}
$$

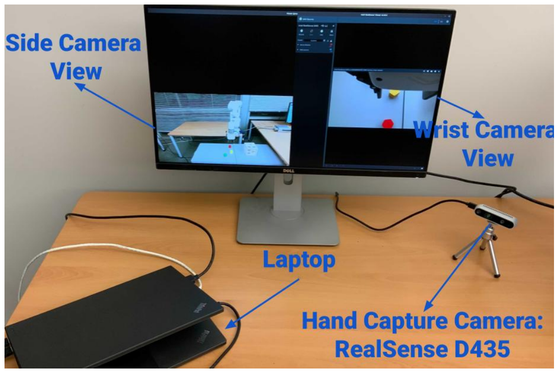
, where $\langle \cdot, \cdot \rangle$ is the inner product operator.

### 6.3.4 Dynamic motion scaling

The scaling factor $\alpha$ determines the distance the end effector moves for each unit of operator movement. A smaller $\alpha$ allows for more precise control of the end effector. For example, if $\alpha = 0.02$, then the end effector will only move $2mm$ if the operator right wrist moves by $10cm$. This allows the operator to precisely control the position of the end effector without having to precisely control their wrist movement. Picking good values for $\alpha$ is thus important to successful task completion. However, good values for $\alpha$ change across tasks and even within one trial of a task depending on task progression. For example, in peg in hole, when the robot has not grasped the peg, a high value of $\alpha$ allows the operator to quickly command the robot to a good pre-grasp pose. When the peg is grasped and ready to be inserted into the hole, smaller values of $\alpha$ are



(a) The scene set-up in robot workspace. The wrist and side cameras stream RGB to a computer screen in the operator space to allow the operator to monitor the motion of the 6DOF arm.



(b) Hardware used by the operator to communicate their intent and observe the robot motion.

**Figure 6.5**: An illustration of the hardware used by our teleoperation system.

preferable.

Thus, when our teleoperation system only controls one robot arm, we use the left hand poses to dynamically adjust the value of $\alpha$ during task execution. If the y coordinate of the left hand wrist with respect to the camera frame is below a certain threshold, we set $\alpha$ to 0.02. Otherwise, we set $\alpha$ to 0.3. In addition, if the distance in image space between the left hand thump and pinky tips is below a threshold, we set $\alpha$ to 0. If $\alpha = 0$, there is no robot motion. Whenever $\alpha$ changes, we reinitialize the origin of the operator frame to be the current right wrist 3d position with respect to the camera frame in the operator workspace. This allows the operator to reset their position when they are almost out of the field of view of the camera. Dynamic motion scaling is an extremely useful yet simple technique that the operator takes advantage of extensively for difficult tasks such as peg in hole. A small value of $\alpha$ also reduces the effect of the noise in hand pose estimates, since pose estimate inaccuracies are also scaled by $\alpha$ during motion correspondence.

### 6.3.5 Hardware system

Figure 6.5 illustrates the hardware used in the robot and operator workspace. The operators observe the current status of the robot workspace, such as the state of the robot and the objects, via a display device in real-time. The RGB-D stream of human motion captured by the operator workspace camera is processed by a laptop to estimate the operator hand poses. The laptop then performs motion correspondence as describe in subsection 6.3.3 to compute the desired robot end effector pose. The desired robot end effector pose is transmitted to the robot controller, which outputs joint torque to achieve the desired pose. The robot controller takes into consideration self-collision and collision with the mounting table, and thus no further motion planning is required. There are two cameras in the robot workspace: a wrist-mounted camera to capture high precision interaction between the grippers and objects, and a tripod-mounted camera to observe the full view of the robot workspace. The RGB video from the two cameras are streamed to the display device in the operator workspace.

## 6.4 Experiments

### 6.4.1 Task descriptions

The key question we investigate is whether given only a single RGB-D camera as the operator motion capture device, can a trained operator teleoperate the robot to perform complex manipulation tasks? To answer this question, we design a set of single arm manipulation tasks in real world or a dual arm coordination task in simulation as shown in Figure 6.1. For the real robot experiments, we consider seven object manipulation tasks: pick and place, peg in hole, hammering, cutting fruit, folding cloth, cord untangling, and object storage in drawer. For the simulated experiment, we consider two-arm manipulation. We describe the tasks in more details below:

**Pick and Place** At the beginning of each episode, three typical objects from the YCB Dataset [CSB+17] are placed in random positions on a table. The robot needs to grasp the object and move at least 0.6 meters to reach the target position. The pick and place serves as the entry level manipulation task.

**Peg in Hole** Three different types of pegs and holes are used, ranked by difficulty from easy to hard: cylinder, pentagon, and hexagon. When the pegs are fully inserted in the holes, the clearance between them is at most 3 mm. We use this task to demonstrate the precision of our tele-operation system since a small position error will lead to task execution failure.

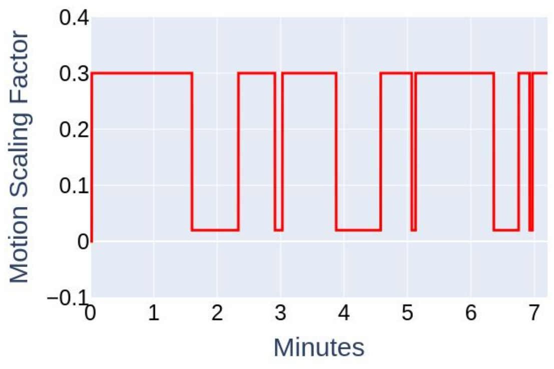
**Hammering** In this task, the robot first picks up a hammer with the gripper. The robot then needs to reach a small bench and hammer the wooden cylinder into the hole. This task evaluates whether the operator can grasp the hammer with a suitable pose and transmit force to the cylinder in the correct direction. Such tool manipulation tasks are of common interest in robotics community [BK19, FZG+20].

**Cutting Fruit** Cutting fruit is another tool manipulation task. The robot uses a knife to slice a watermelon chunk into two pieces. Similar to hammering, cutting requires the robot to grasp the knife with the right pose and transmit force in the correct direction to the watermelon chunk.

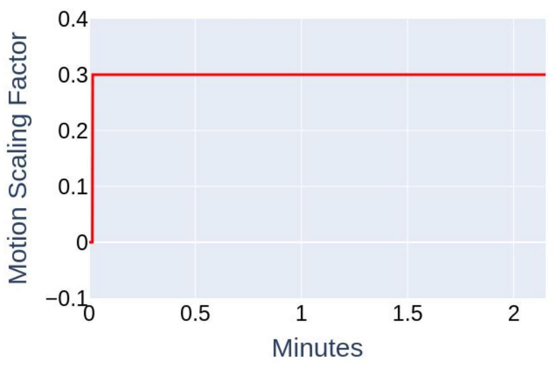
**Folding Cloth** A piece of cloth lies flat on a table. The robot folds the cloth twice along the diagonal. This task demonstrates deformable object manipulation ability.

**Moving Large Container** Two robot arms move a large container from one planar surface to another. Due to a lack of hardware, we demonstrate the successful completion of this task by our system in the simulator DRAKE [TtDDT19]. This task tests the coordination of two arms where each hand of the operator controls each robot arm. To control the second robot arm, we apply the same motion correspondence procedure described in subsection 6.3.3 to the left hand wrist.

**Cord Untangling** One end of a cord is fixed. The remaining cord section is tangled to make a knot. The robot untangles the cord by grasping and pulling the cord in the right direction.



(a) Peg in Hole.



(b) Pick and Place.

**Figure 6.6**: The x-axis in both figures is the trial duration. The y-axis is the dynamic translational motion scaling $\alpha$. **Top:** For peg in hole, the operator switches back and forth between large and small $\alpha$ values depending on current task progress **Bottom:** For simple pick and place, the operator can finish the task with a constant high scaling factor.

**Object Storage in Drawer** This task requires executing four primitive manipulation skills: pull the drawer to open, pick an object, place it into the drawer, and close the drawer. The task also requires manipulating articulated object, a problem which recently receives significant attention [UHC+19, XQM+20].

For each tasks, the operator performs 3 evaluation trials after a period of practice, except for *Pick and Place* where we perform 5 trials and *Moving Large Container* where we only perform 1 trial.

We highlight that the *Cord Untangling* and *Object Storage in Drawer* tasks were not seen by the operator before evaluation. The first time the operator interacts with the objects present in these two tasks is during evaluation. Such unseen tasks allow us to test the adaptability of the system and the operator to new tasks and operating conditions.

### 6.4.2 Results of using RGB-D camera to capture hand poses

For all the tasks, our teleoperation system successfully completes the evaluation trials. This fact is even more surprising for the *Cord Untangling* and *Object Storage in Drawer* tasks where the operator has not practiced with the objects before the evaluation trials. Such successes

| Tasks | Pick and Place | Peg in Hole | Hammering | Cutting Fruit | Folding Cloth | Object Storage in Drawer | Cord Untangling |
|---|---|---|---|---|---|---|---|
| Time | $8.9 \pm 1.3$ | $9.7 \pm 0.8$ | $3.8 \pm 0.8$ | $3.9 \pm 0.1$ | $5.7 \pm 0.7$ | $7.0 \pm 1.2$ | $4.4 \pm 1.1$ |

**Table 6.1**: Time to complete each task (in minutes, $\pm$ indicate std). Given desired end effector pose, our teleoperation system uses the controller provided by the robot manufacturer to control joint torque. The controller has high latency, leading to high task time-to-completion. Reducing the controller latency will significantly reduces the time-to-completion.

on unseen tasks demonstrate the generality of our teleoperation system and can potentially allow human operators to remotely control robot *in the wild* in unseen conditions. The time to completion for each tasks are shown in Table 6.1.

We notice interesting failure recovery behavior. In the first trial of the *Cord Untangling* task, the end effector became stuck under a table in the robot workspace. Without manual reset of the robot, the operator was able to unstuck the end effector by moving under the table and eventually complete the task. A common failure mode is closing the gripper without grasping the objects. This happens more frequently with small objects since neither the wrist camera or the side camera in the robot space provide distinctive visual cues to the operator whether the object is between the two gripper pads. However, this failure mode is easy to recover from because the operator can retry the grasp.

In addition to failure recovery, the operator also discovers interesting manipulation strategy such as dynamic manipulation. In the *Cord Untangling* task when the end of the cord (with higher mass density) is stuck underneath itself, the operator quickly moves their wrist to induce a large position error, thus inducing a large acceleration of the robot, which unstucks the end of the cord due to inertia.

To ablate the benefit of the dynamic motion scaling scheme introduced in subsection 6.3.4, the operator performs the peg in hole task while using a constant translational scaling factor $\alpha$. For a range of $\alpha$ values in $0.03, 0.3, 0.5, 1.0$, the operator was not able to successfully accomplish the task when given 15 minutes to teleoperate the robot. For $\alpha = 0.03$, the operator can not command the robot to reach the peg before going out of the field of view of the operator space camera. For higher values of $\alpha \in 0.3, 0.5, 1.0$, the operator fails to control the robot precisely enough to insert the peg into the hole. The ability to dynamically modify the scaling factor $\alpha$ is most beneficial for precision manipulation tasks and less important for simple task, as illustrated in Figure 6.6.

### 6.4.3 Results of using RGB camera to capture hand poses

Using only RGB to capture hand pose leads to successful completion of 1 out of 3 trials in the pick and place task. In the first two trials, the operator can grasp and move the three objects, but ends up knocking one of the objects over due to noisy depth estimates. The main failure modes of pose estimation occur when then palm plane is not parallel to the image plane or when the right hand index and thump tips come close together to signal that the gripper should close. These situations violate the weak perspective assumption and leads to a non-trivial drop in the estimated depth accuracy. After becoming accustomed to the failure modes of the systems, the operator successfully completes the third trials.

## 6.5 Conclusions

We present a teleoperation system for a 6-DOF robot arm that uses a single RGB-D camera to capture the operator intent. The operator actions are mapped to robot coordinates. We demonstrate successful teleoperation for a broad set of tasks such as cutting, hammering, peg-in-hole, folding deformable materials, etc. Interesting future works include dual arm manipulation, performing more complex tasks using only RGB sensing and teleoperating multi-finger robot hand.

## 6.6 Acknowledgement

Chapter 6, in part, is a reprint of material from Quan Vuong, Yuzhe Qin, Runlin Guo, Xiaolong Wang, Hao Su, Henrik Christensen. “Single RGB-D Camera Teleoperation for General Robotic Manipulation”. The dissertation author is the primary author of the paper.

# Chapter 7

# How to pick the domain randomization parameters for sim-to-real transfer of reinforcement learning policies?

## 7.1 Introduction

Recently, reinforcement learning (RL) algorithms have demonstrated remarkable success in learning complicated behaviors from minimally processed input ([HZAL18a, VZR18, SLM$^+$15, FvHM18, BHB$^+$18, CPPH$^+$19]). However, most of this success is limited to simulation. While there are promising successes in applying RL algorithms directly on real systems ([ZVS$^+$18, MKV$^+$18, CFFF19, FRF$^+$18, TZC$^+$18]), their performance on more complex systems remains bottle-necked by the relative data inefficiency of RL algorithms. Domain randomization is a promising direction of research that has demonstrated impressive results using RL algorithms to control real robots ([TFR$^+$17, PAZA17, YLT18, OAB$^+$18, JWK$^+$18]).

At a high level, domain randomization works by training a policy on a distribution of environmental conditions in simulation. If the environments are diverse enough, then the policy trained on this distribution will plausibly generalize to the real world. A human-specified design choice in domain randomization is the form and parameters of the distribution of simulated environments. It is unclear how to the best pick the form and parameters of this distribution and prior work uses hand-tuned distributions. This extended abstract demonstrates that the choice of the distribution plays a major role in the performance of the trained policies in the real world and that the parameter of this distribution can be optimized to maximize the performance of the trained policies in the real world.[1]

## 7.2 Background and Notation

In RL, the robotic learning problem is abstracted as a discrete time sequential decision making problem in a Markov decision process (MDP). An MDP is a tuple $(\mathcal{S}, \mathcal{A}, r, T, \gamma, \rho)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, state transition function $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, a discount factor $\gamma$ and a distribution over the initial state $\rho$. Given a state $s \in \mathcal{S}$, a policy $\pi_\theta$ defines a distribution $\pi_\theta(.|s)$ over the action space $\mathcal{A}$. $\theta$ represents the parameters of the policy, which can be linear operators ([RLTK17b, MGR18]) or the weights and biases of a deep neural network. Let $m$ denotes one specific MDP $(S^{(m)}, A^{(m)}, r^{(m)}, T^{(m)}, \gamma, \rho^{(m)})$. The performance of a policy $\pi_\theta$ with respect to the MDP parameterized by $m$ is evaluated by:

$$J^{(m)}(\pi_\theta) \triangleq \mathbb{E}\tau \sim \pi_\theta \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$$

where $\tau = (s_0, a_0, r_0, s_1, \ldots)$ is a trajectory generated by using the policy $\pi_\theta$ to interact with the MDP $m$. Let $m_{\text{real}}$ denotes the MDP representing the real world. Formally, domain randomization

[1]Dockerized code to reproduce our experiments is available at https://github.com/quanvuong/domain_randomization.

performs the optimization

$$\theta^* = \arg\max_{\theta} \mathbb{E} m \sim p_{\phi} J^{(m)}(\pi_{\theta})$$

where $p_{\phi}$ is a distribution over MDPs parameterized by $\phi$. $\pi_{\theta^*}$ is then used to perform the task of interest in $m_{\text{real}}$. For example, in [PAZA17] where domain randomization was successfully used to transfer a policy trained in simulation to the real world on object pushing tasks, $\phi$ parameterizes the distribution over the masses and damping coefficients of the robot's links in addition to other environmental conditions.

## 7.3 Optimization of the domain randomization parameters

At a high level, domain randomization is a technique to accomplish the general goal: "Given a simulator, we want to use it such that when we train a policy in the simulator, the policy will perform well in the real world". We argue that this is an objective that we can optimize for directly. In prior works, the parameter $\phi$ of the distribution over MDPs is chosen by hand, presumably using domain knowledge and through trial-and-error; it is also kept fixed throughout the training process. Prior works also assume that there is a clear demarcation between training and testing, i.e. during training in simulation, the policy does not have access to the real system. However, in practice, this assumption could be incorrect, as we may have limited or costly access to the real system. In such scenarios, we could use the real system to provide some signal for domain randomization.

With access to the real environment $m_{\text{real}}$, we can formalize domain randomization as a bilevel optimization problem:

$$\arg\max_{\phi} \quad J^{(m_{\text{real}})}(\pi_{\theta^*(\phi)}) \tag{7.1}$$

$$\text{such that} \quad \theta^*(\phi) = \arg\max_{\theta} \mathbb{E} m \sim p_\phi J^{(m)}(\pi_\theta) \tag{7.2}$$

To establish that this is a research direction worth pursuing, we need to demonstrate the following:

- The choice of the parameter $\phi$ plays a major role in the performance of the policies in the real environment.
- $\phi$ can be optimized to increase the performance of the trained policies in the real environment.

We experimentally demonstrate these two points by using Cross Entropy Method (CEM) to approximately solve the outer problem (Equation 7.1) and Proximal Policy Optimization (PPO) [SWD+17] to solve the inner problem (Equation 7.2). The closest related work to ours is [CHM+18b], which finds the simulation parameters that bring the state distribution in simulation close to the state distribution in the real world. We argue that this is only a proxy measure of the actual objective we ultimately care about and optimize for directly, i.e. the performance of the trained policy in the real environment. Other than domain randomization, other parallel research directions for sim-to-real transfer exist and have been demonstrated to be promising research areas as well ([HLD+19, YKTL19, ICT+18, ZGB+17, TZC+18]).

## 7.4 Algorithmic Description

CEM is a simple iterative gradient-free stochastic optimization method. Given the decision variable $\phi$, CEM alternatives between evaluating its current value on the objective function

(Equation 7.1) and updating $\phi$. We refer interested readers to [dBKMR] for a more detailed description. We initialize $\phi$ with $\phi_0$, evaluate $\phi_0$ to obtain $J^{(m_{\text{real}})}(\pi_{\theta^*(\phi_0)})$, use the evaluation result to update $\phi_0$ to obtain $\phi_1$, and so on.

## 7.5 Experimental Settings and Results

To demonstrate the potential of our research direction, we focus on transferring learned policies between two simulators. Specifically, we focus on transferring policies for the environments Hopper and Walker from the Dart simulator [LGH+18] to the Mujoco simulator [TET12]. Thus, $m_{\text{real}}$ represents the parameters of the MDP in the Mujoco simulator. Transferring between these two simulators has been demonstrated to be a fruitful experimental testbed for sim-to-real studies [YLT18]. In our setting, the MDPs in both simulators are parameterized by the masses, damping coefficients of the robot's links and the gravity constant ($\mathbb{R}^9$ for Hopper and $\mathbb{R}^{15}$ for Walker).

$\phi$ represents the parameters of a distribution over $m$. In our experiments, $\phi$ is the mean and variance of a diagonal Gaussian distribution over the simulation parameters. The initial mean $\phi_0$ is set to $m_{\text{dart}}$ and initial variance set to 1 for all parameters. These values are reasonable defaults for the domain randomization distribution parameters without domain knowledge or trial-and-error. CEM is then used to optimize for $\phi$.

We replicate our results for each setting over 5 different random seeds. In the Hopper environment, the performance of the policies trained with the optimized $\phi$ is on averaged 102% higher than the performance of the policy trained with the initial value $\phi_0$ with a standard deviation of 48% and minimum improvement of 28%. In the Walker environment, the performance of the policies trained with the optimized $\phi$ is on average 80% higher than the performance of the policy trained with the initial value $\phi_0$ with a standard deviation of 53% and minimum improvement of 19%. The existence of a better value for $\phi$ than $\phi_0$ shows that environment distributions chosen by hand can be improved with optimization. Furthermore, our result is consistent with ongoing research in domain randomization for supervised learning which demonstrated the importance of the sampling distribution for sim-to-real transfer success ([RSC18, PBB+18]).

## 7.6 Future Research Directions

### 7.6.1 Learning complex distributions

We assume a diagonal Gaussian sampling distribution for simplicity, but learning a more complex distribution could result in a better randomized environments. For example, deep generative modeling approaches such as variational autoencoders ([KW13, RMW14]) and autoregressive flows ([PPM17, KSJ+16]) could be used to model complex dependencies and correlations between simulation parameters.

### 7.6.2 Optimization techniques

CEM was chosen to solve the outer problem (7.1) due to its simplicity. We are interested in more advanced gradient-free optimization methods, such as CMA [LH16] or Bayesian optimization [KVN+19]. If we assume that the parameter $\phi$ is parameterized by a distribution $p_\omega$, it can be shown that $\nabla_\omega \underset{\phi \sim p_\omega}{E}[J^{(m_{\text{real}})}(\pi_{\theta^*(\phi)})] = \underset{\phi \sim p_\omega}{E}[\nabla_\omega \log p_\omega(\phi) J^{(m_{\text{real}})}(\pi_{\theta^*(\phi)})]$ and we could apply stochastic gradient-based techniques to directly optimize for $\omega$. We are also particularly excited about asynchronous evolutionary algorithms (AEA). Whereas previous techniques are synchronous by nature, AEA enables simultaneously training policies in simulation and evaluating in reality, thereby making the best use of the available resources in terms of wall-clock time.

### 7.6.3 Off-policy Reinforcement Learning

PPO, an on-policy RL algorithm, was chosen to solve the inner problem (7.2) due to its simplicity and speed. However, off-policy training of the policy with real world data has been demonstrated to improve the policy performance ([JWK$^+$18, GHLL16]). In our setting, the real world data generated to evaluate the policy at every iteration of solving the outer problem (7.1) can be used to optimize the next policy in an off-policy fashion. Preferably, the inner problem (7.2) is solved by an off-policy algorithm to allow for easy fine-tuning of the trained policy on real world data.

### 7.6.4 Transferable Domain Randomization Parameters and Testing On Real Robots

It would be of interest to understand if there exists general principles to determine the value of $\phi$ or transferable initial values for $\phi$ that works for domain randomization across a wide range of tasks and robots. This is so that the expensive problem (Equation 7.1 and 7.2) does not have to be solved from scratch for every problem instance. Ultimately, the goal is to test our approach to domain randomization on real robots.

## 7.7 Acknowledgement

Chapter 7, in part, is a reprint of material from Quan Vuong, Sharad Vikram, Hao Su, Sicun Gao, Henrik I. Christensen. “How to pick the domain randomization parameters for sim-to-real transfer of reinforcement learning policies?”. In IEEE International Conference on Robotics and Automation Learning Legged Locomotion Workshop, 2019. The dissertation author is the primary author of the paper.

# Chapter 8

# Automatically reconstructing scenes to train grasping network

## 8.1 Introduction

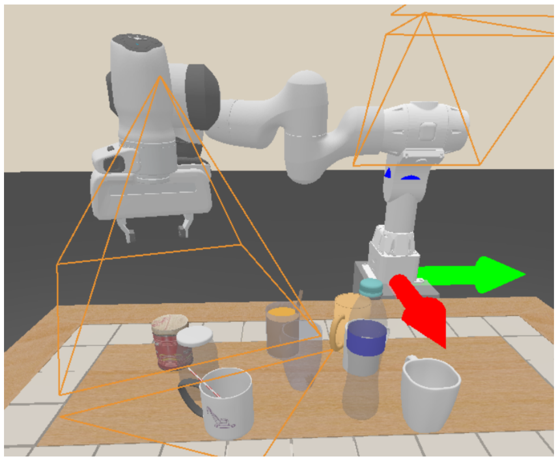


**Figure 8.1**: A visualization of our test scene. Given objects placed on a tabletop surface, a robot should pick up the objects and move them above the tabletop surface.

Learning robotic manipulation skills in simulation and executing the learnt skills in the real world, often termed *Sim2Real*, has fueled many recent advances in robot manipulation [MLN$^+$17, MPH$^+$16, MML$^+$17, HRX$^+$20, XCB$^+$22]. These approaches usually require practitioners to manually curate the object meshes, place them at realistic poses in the simulation scenes and calibrate their dynamics parameters. The process of manual scene creation and calibration requires domain expertise and can be prohibitively costly to scale to large-scale scenes with many objects. Perhaps this is one of the reasons why applications using Sim2Real have mostly been demonstrated on manipulation tasks in constrained settings involving a single object, such as rope manipulation, or when the simulated scenes can be procedurally generated, such as in bin picking. Recognizing scene creation and calibration as a major bottleneck of Sim2Real, recent research has attempted to automate this process and dub the problem *Real2Sim2Real* [LHC+21].

In fact, we can refer to these recent research as *dynamics Real2Sim2Real* since they have mostly focused on estimating the dynamic parameter of the physics simulation [CHM+18a, DWD+21]. Relatively less attention has been paid to the challenge of *geometric Real2Sim2Real* – automatically constructing the geometry of the objects in the scenes and placing them at realistic poses that allow for forward simulation. Constructing object meshes and placing them into simulated scenes in a way that represents the distributions of objects in the real world remains a highly manual process [KMH+17, LXMM+21, SCU+21]. In contrast, research in autonomous driving has benefited tremendously from efforts to automatically reconstruct virtual clones that mimic the realistic diversity captured in the real world [KPL+19, DKF20, GWCV16, CMH20, IKK+22]. These virtual clones of outdoor driving scenes have allowed for the collection of large-scale ground truth labels for vision tasks [CMH20], simulation of pedestrian behaviors [IKK+22], and even training driving policy end-to-end inside simulation [BRL+18]. However, these virtual clones stop short of allowing for forward simulation of dynamics, which is often necessary for application in robotic manipulation. Can we create virtual clones of the real world that allow for forward simulation? Will these virtual clones be useful for manipulation research? Are there existing methods that tackle this challenge? What are their strengths and weaknesses?

To answer these questions, we develop a benchmark for *geometric Real2Sim2Real*. We use the task of grasping in clutter as a testbed to study different algorithms. The benchmark consists of 30 testing scenes. Figure 8.1 provides a visualization of the test scene. Each scene comprises objects resting on a tabletop surface and a six degree of freedom robot arm placed next to the table. Given observations of the scene captured from vision sensors, an algorithm must first create a clone of the scene in a simulation instance (the *Real2Sim step*). That is, for each object in the scene, the algorithm should generate a mesh for the object and place the mesh in the simulation environment near the ground truth pose of the object. After this step, the algorithm uses the reconstructed scene to find successful grasps for objects in the real scene (the *Sim2Real step*).

Several recent works attempt to automate the reconstruction of scene geometry in the forms of object meshes [HZJ+21, JHZ22]. However, in these prior works, the reconstructed scenes have not been used to learn robotics skills. We demonstrate that neural networks that learn to output implicit representation can reconstruct the test scene in an automated fashion with excellent results. We demonstrate that the reconstructed scenes allow training a recent state-of-the-art grasping network. The trained network outperforms a publicly available grasping model, with the same architecture but different weights. Most notably, the publicly available models were trained using 17.7 million simulated grasp labels. In contrast, the same architecture obtains higher test performance while using only the reconstructions of the 30 test scenes and thousands of grasp labels for training.

## 8.2 Problem Statement

We test our approach on the task of grasping objects in clutter using 6 degree-of-freedom grasp. There are 30 test scenes. In each test scene, objects are placed on a table top next to a robot arm equipped with a parallel yaw gripper. The objects belong to four categories: bottle, bowl, can, and mug. Bottles are articulated models collected from PartNet-Mobility [XQM+20], while the rest of the categories are rigid models from ShapeNetCorev2 [CFG+15]. Each scene consists of

5 to 10 randomly chosen objects. To place the objects on the table, we use random object poses while ensuring that the objects are standing upright, there are no intersection between the objects, and there are no object stacked on top of another object.

In each test scene, depth cameras are placed above the table and their poses are set such that they look at the center of the table. Given the depth map captured by each camera, an algorithm should reconstruct the test scene in simulation and use the test scene to train a grasping network. The performance of the reconstruction algorithm is then evaluated by the performance of the trained grasping network, when evaluated in the test scene on the task of grasping object using 6 degree-of-freedom grasp. To ensure realistic sensor noise in the captured depth map, we use SimKinect to add noise to the noise-free depth map captured by the cameras [HWMD14, BM13, BRHS14].

## 8.3 Method

We provide description of the reconstruction framework in subsection 8.3.1, the grasping algorithm in subsection 8.3.2 and implementation details in subsection 8.3.3.

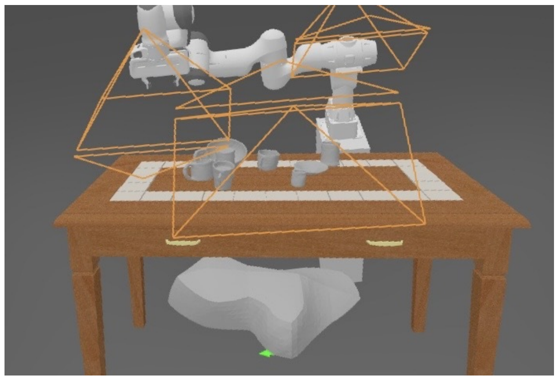
### 8.3.1 Reconstruction framework

Our reconstruction framework takes as input depth maps captured by the cameras of the test scenes and produces object meshes and their poses. The object meshes and their poses allow us to place the objects in a simulation environment, which is subsequently used to train the grasping network. The first step of our framework consists of converting the depth maps into point clouds and then fusing the point clouds corresponding to different cameras in the scene into a single scene-level point cloud. We then use object segmentation to segment the scene-level point cloud into object-level point clouds. That is, each object-level point cloud only contains points that lie on the surface of the same object instance. Given an object-level point cloud as input, we use a trained Convolutional Occupancy Network (ConvONet) [PNM$^+$20] to produce an implicit representation of the object surface. The object mesh can then be obtained from the implicit representation by the Multiresolution Isosurface Extraction procedure introduced in [MON$^+$18].

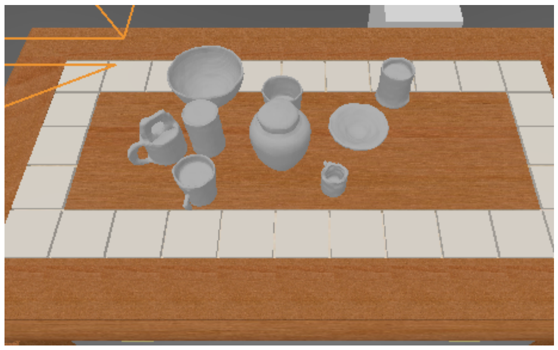
We next describe how we place the reconstructed object meshes into the scene. Given an object-level point cloud, represented with respect to the world frame, centering and scaling operations are applied to the point cloud before the point cloud is inputted into the ConvONet. The parameters of the centering and scaling operations are computed from the object-level point cloud such that the resulting point cloud center coincides with the origin of the world frame. Additionally, after the scaling operation, the largest edge length of the axis-aligned bounding box of the point cloud should be 1. Given the reconstructed object mesh corresponding to one object-level point cloud, we apply the inverse of the centering and scaling operations to the object mesh to place the object mesh into the simulation scene. In other words, the inverse of the centering and scaling operations can be interpreted as the object pose.

Algorithm 5 provides the step-by-step description of our reconstruction framework. In addition to the steps discussed above, we apply two additional steps to improve the reconstruction quality. Firstly, we use statistical point cloud outlier removal [ZPK18] to remove outlier points in the object-level point cloud before computing the parameters for the centering and scaling operations. This is because these two operations are sensitive to noise in the observed point cloud. Figure 8.2 provides an example that illustrates the benefit of this step. Secondly, given the object mesh constructed by the Multiresolution Isosurface Extraction (MISE) procedure, we perform approximate convex decomposition to obtain a simplified and smoother object mesh representation [Mam16], since the mesh obtained by MISE often has complex geometry, leading to slow collision detection during simulation.

To train the ConvONet, we generate the training data from 80 training scenes. The training scenes are generated using the same methodology we use to generate the test scenes. However, the objects used to generate the training and test scenes form disjoint sets, i.e. no object mesh



(a) Visualization of a reconstructed scene without applying point cloud outlier removal to the observed point cloud. The outliers in the point cloud drastically affect the centering and scaling operations, leading to incorrect reconstructed shape (the big blob under the table).



(b) Visualization of the reconstruction of the same scene with point cloud outlier removal applied to the observed point cloud. The previously erroneously reconstructed mesh shown in the left figure now has more accurate reconstruction.

**Figure 8.2**: The figure illustrates the importance of applying point cloud outlier removal to the observed point cloud to ensure the quality of the reconstructed meshes.

used to generate the training data for ConvONet exist in the test scenes and vice versa. For each object in each training scene, we obtain the object-level point cloud by fusing multi-view depth maps. The object-level point clouds are the input into ConvONet during training. To obtain the ground-truth annotation, we use ground-truth occupancy labels. We train ConvONet using the binary cross entropy loss. We refer interested readers to [PNM+20, MON+18, JHZ22] for more details on the training details of ConvONet.

In our reconstruction framework, we assume that we have access to the ground truth mesh and pose of the table in the test scene. The reconstruction algorithm, therefore, does not have to reconstruct the table. We also assume access to the ground truth semantic segmentation when reconstructing the test scenes.

### 8.3.2 Grasping Network

Given the reconstructed scene obtained from the framework described in subsection 8.3.1, we measure the benefit of the reconstruction depending on whether the reconstruction allows

**Algorithm 5** Step-by-step description of our reconstruction framework to convert depth maps and camera poses to a simulation environment

```
    Input: $N$ depth maps, semantic segmentation maps and camera poses
    Output: A simulation environment
 1: Convert the depth maps to point clouds and fuse the point clouds
 2: Extract object-level point clouds using the semantic segmentation maps
 3: for Each object-level point cloud do
 4:     Perform point cloud outlier removal
 5:     Compute the centering and scaling operation $T \in SE(3)$
 6:     Apply $T$ to the point cloud
 7:     Use a trained ConvONet to obtain an implicit representation from the point cloud
 8:     Use Multiresolution Isosurface Extraction to obtain object mesh from the implicit representation
 9:     Perform approximate convex decomposition to the object mesh
10:     Set the object pose to be $T^{-1}$
11:     Place the object mesh into the simulation scene using $T^{-1}$
12: end for
```

us to train neural network. This is different from the standard metrics often used in the vision community, which only measures how well the reconstructed shape approximates the ground truth shape, such as the Chamfer distance. More specifically, we use the reconstructions of the test scenes to generate ground truth grasp pose and grasp success label to train a grasping network. We then use the grasp success of the trained network when evaluated on the test scenes to judge whether the reconstruction is of high quality. In our work, we use Contact-GraspNet (CGN) [SMTF21] as the grasping network architecture. CGN takes as input as point cloud and predicts for every point in the point cloud a corresponding grasp pose and the probability of grasp success.

We pick CGN over existing grasping network architecture for a few reason. The network takes as input point cloud, and does not require color information as input. Such input requirement is suitable for our reconstruction because our reconstruction framework only reconstructs the shape and not the color of the object. We therefore only have access to the depth maps of the scenes and not color images. CGN is also a recent state-of-the-art grasping network with excellent reported performance. Last but not least, a pre-trained CGN model is released by the original author. The model was trained using 17.7 million simulated grasp. The pretrained-model serves as a strong baseline for us to compare the performance of the model trained using our reconstructions to.

### 8.3.3 Implementation Details

To perform grasping, we use a panda robot arm with a fixed base. Given a desired pose of the gripper, we use the MPlib library [MG21] to plan a trajectory in joint space the moves the robot gripper from the current to the desired pose while avoiding collision. We use a parallel-yaw gripper and the gripper width is always set to a predefined width before executing the grasp.

## 8.4 Experimental Results

| Grasp Proposal Network | Grasp success averaged over 30 test scenes |
|---|---|
| CGN trained on 30 reconstructed scenes | 0.756 |
| Pre-trained CGN (trained with 17.7 million grasps) | 0.68 |

**Table 8.1**: The table demonstrates the quality of the reconstructed scene. The CGN model trained using our reconstructed scenes outperforms the publicly available model by 11%, even though the publicly available model was trained using 17.7 million grasp labels.

Table 8.1 illustrates the quality of our reconstructed scenes, in the sense that the CGN model trained using our reconstructed scenes outperforms the publicly available model by 11%. This is a particularly exciting result because the publicly released model was trained using 17.7 million simulated grasps. Even more importantly, these grasps were taken from Acronym, a large-scale grasp dataset [EMF20]. The corresponding object meshes used in the Acronym dataset are also placed on a tabletop surface using random poses, a training setting very similar to our test conditions. This ensures that the pre-trained model does not suffer from obvious generalization challenges when evaluated on our test scenes. We, therefore, conclude that the reconstructed scenes allow the trained CGN model to generalize better to the test scenes, leading to improved performance compared to the randomly generated scenes used to train the pre-trained model.

| | Grasp success averaged over 30 test scenes |
|---|---|
| Determining grasp label using mean success rate | 0.756 |
| Determining grasp label using one execution | 0.716 |

**Table 8.2**: The table demonstrates the importance of determining grasp success label by executing the grasp multiple times, and only label as successful grasp pose whose mean success rate is above a pre-defined threshold.

We also found that to obtain good performance, it was important to determine the grasp success label by executing the grasp multiple times, and only label it as a successful grasp pose when the mean success rate is above a pre-defined threshold. Table 8.2 illustrates quantitatively the importance of doing so, measured by the grasp success rate of the CGN model trained using grasp pose labels generated from the reconstructed scene.

## 8.5 Acknowledgement

Chapter 8, in part, is a reprint of material from Hao Su, Henrik Christensen, Luobin Wang, Quan Vuong, Runlin Guo, Yuzhe Qin. Ongoing work in the research topic of geometric real2sim2real. Authors listed in alphabetical order. The dissertation author is the primary investigator.

# Chapter 9

# Conclusion

This thesis describes our work in areas key to developing agents that can autonomously acquire skills in physical environments. In this chapter, we discuss limitations and potential areas for future work.

Developing robust Reinforcement Learning (RL) algorithm that can learn well from data distribution with different properties while requiring minimal tuning effort remains an extremely challenging problem. While this thesis introduces novel RL algorithms and demonstrates their performance on standard benchmarks commonly used by researchers, the potential for these algorithms has yet to be explored on a wide range of real-world robotics problems. Perhaps what is missing is a tried-and-true recipe for applying RL to real-world problems, a comprehensive guide on how to analyze the performance of the trained policies and various metrics for effective debugging, and a theory that accurately and precisely predicts under what conditions would applying RL yields the desirable level of performance.

The RL algorithms develop in this thesis also learn exclusively from interaction data, which can be hard to collect in practice. Humans can learn from more diverse data sources, such as by observing other humans' behavior, reading instructions in textual format, or more generally by processing data not directly experienced by oneself. Endowing RL agents with the ability to learn from more diverse sources of data, such as video or text, can be a highly impactful endeavor that improves generalization and learning efficiency.

Yet another limitation is the task-centric perspective of the proposed algorithms. That is, a task needs to be manually defined by the algorithm designer before the agent can start learning. Even if the agent masters this specific task, it does not generalize in a zero-shot manner to other tasks. Learning to perform another task often requires another expensive round of data collection and training. As such, the task-centric nature of the proposed algorithms seems hard to scale to flexible agents that can perform many useful tasks. While our work in chapter 2 demonstrates initial promise in training RL agents that can perform many tasks, the algorithm still relies on a manually defined task distribution. We are very interested in continuing to develop RL agents that define their tasks to practice on their own, learn many tasks and demonstrate zero-shot capability to tasks not seen during training.

Switching gear, we believe that developing novel interfaces to collect human supervision and demonstration will play a key role in scaling up RL agents for robotic manipulation. Our system presented in chapter 6 can allow the teleoperator to perform complex tasks. But the time taken for each task tends towards to higher range due to the latency of the robot controller. Reducing the response time of the robot controller, or even developing new controllers that can preemptively move before the teleoperator provides explicit commands are interesting directions for future work.

Last but not least, to the best of our knowledge, our work in chapter 8 is the first to demonstrate that we can improve the performance of neural networks when they are trained using object meshes generated automatically by another neural network. We believe this is very exciting progress, and we look forward to the days when there are plenty of large-scale reconstructions of diverse real-world environments that allow for forward simulation. These reconstructions may serve as a playground for RL agents to hone their skills before being released into the real world, as we have discussed previously as another avenue for future work. The most obvious limitation of our current work is that we do not reconstruct the color information of the objects in the scenes, only their geometric shape. It would be interesting to apply recent advances in recovering color information of objects from sensor data, such as Neural Radiance Fields [MST$^{+}$20], and test whether the resulting reconstructed scenes allow for training RL agents.

# Bibliography

[ABC$^+$] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation.

[Ach] Joshua Achiam. Advanced policy gradient methods. http://rll.berkeley.edu/deeprlcourse/f17docs/lecture_13_advanced_pg.pdf.

[AGHK18] Dafni Antotsiou, Guillermo Garcia-Hernando, and Tae-Kyun Kim. Task-oriented hand motion retargeting for dexterous manipulation imitation. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[AHTA17] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 22–31. JMLR. org, 2017.

[ALP$^+$15] Abbas Abdolmaleki, Rudolf Lioutikov, Jan R Peters, Nuno Lau, Luis Pualo Reis, and Gerhard Neumann. Model-Based Relative Entropy Stochastic Search. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3537–3545. Curran Associates, Inc., 2015.

[Ama98] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[ANAA16] Riad Akrour, Gerhard Neumann, Hany Abdulsamad, and Abbas Abdolmaleki. Model-Free Trajectory Optimization for Reinforcement Learning. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2961–2970. JMLR.org, 2016.

[ASN19] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. *arXiv preprint arXiv:1907.04543*, 2019.

[AST$^+$18a] A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations (ICLR)*, 2018.

[AST$^+$18b] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Rémi Munos, Nicolas Heess, and Martin A. Riedmiller. Maximum a Posteriori Policy Optimisation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[BB00] J. Baxter and P. L. Bartlett. Direct gradient-based reinforcement learning. In *IEEE International Symposium on Circuits and Systems, ISCAS 2000, Emerging Technologies for the 21st Century, Geneva, Switzerland, 28-31 May 2000, Proceedings*, pages 271–274. IEEE, 2000.

[BB01] Jonathan Baxter and Peter L. Bartlett. Infinite-Horizon Policy-Gradient Estimation. *J. Artif. Intell. Res.*, 15:319–350, 2001.

[BBT19] Adnane Boukhayma, Rodrigo de Bem, and Philip HS Torr. 3d hand shape and pose from images in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10843–10852, 2019.

[BBW01] Jonathan Baxter, Peter L. Bartlett, and Lex Weaver. Experiments with Infinite-Horizon, Policy-Gradient Estimation. *J. Artif. Intell. Res.*, 15:351–381, 2001.

[BCP$^+$16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

[BDR$^+$20] Yoshua Bengio, Tristan Deleu, Nasim Rahaman, Nan Rosemary Ke, Sebastien Lachapelle, Olexa Bilaniuk, Anirudh Goyal, and Christopher Pal. A meta-transfer objective for learning to disentangle causal mechanisms. In *International Conference on Learning Representations*, 2020.

[Bel57] Richard Bellman. *Dynamic Programming*. Dover Publications, 1957.

[BHB$^+$18] Gabriel Barth-Maron, Matthew W. Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, and Timothy P. Lillicrap. Distributed distributional deterministic policy gradients. *CoRR*, abs/1804.08617, 2018.

[BK19] Aude Billard and Danica Kragic. Trends and challenges in robot manipulation. *Science*, 364(6446), 2019.

[BM13] Jonathan T. Barron and Jitendra Malik. Intrinsic scene properties from a single rgb-d image. *CVPR*, 2013.

[BNVB12] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *CoRR*, abs/1207.4708, 2012.

[BRHS14] Jeannette Bohg, Javier Romero, Alexander Herzog, and Stefan Schaal. Robot arm pose estimation through pixel-wise part classification. *ICRA*, 2014.

[BRL$^+$18] Alex Bewley, Jessica Rigley, Yuxuan Liu, Jeffrey Hawke, Richard Shen, Vinh-Dieu Lam, and Alex Kendall. Learning to drive from simulation without real world labels, 2018.

[Bro91a] Rodney A. Brooks. Intelligence without reason. In *COMPUTERS AND THOUGHT, IJCAI-91*, pages 569–595. Morgan Kaufmann, 1991.

[Bro91b] Rodney A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47(1):139–159, 1991.

[BRS04] Jurgen Broeren, Martin Rydmark, and Katharina Stibrant Sunnerhagen. Virtual reality and haptics as a training device for movement rehabilitation after stroke: a single-case study. *Archives of physical medicine and rehabilitation*, 85(8):1247–1250, 2004.

[BT02] Ronen I. Brafman and Moshe Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[BWRB21] David Brandfonbrener, William F Whitney, Rajesh Ranganath, and Joan Bruna. Offline rl without off-policy evaluation. *arXiv preprint arXiv:2106.08909*, 2021.

[Cal10] James J. Callahan. *Advanced Calculus: A Geometric View*. Springer Science & Business Media, September 2010.

[CCML18] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 4759–4770, 2018.

[CCN$^+$20] Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, Mel Vecerík, Oleg Sushkov, David J. P. Barker, Jonathan Scholz, Misha Denil, Nando de Freitas, and Ziyu Wang. Scaling data-driven robotics with reward sketching and batch reinforcement learning. *arXiv: Robotics*, 2020.

[CFFF19] H. L. Chiang, A. Faust, M. Fiser, and A. Francis. Learning navigation behaviors end-to-end with autorl. *IEEE Robotics and Automation Letters*, 4(2):2007–2014, April 2019.

[CFG$^+$15] Angel Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. 12 2015.

[CGY$^+$22] Xi Chen, Ali Ghadirzadeh, Tianhe Yu, Yuan Gao, Jianhao Wang, Wenzhe Li, Bin Liang, Chelsea Finn, and Chongjie Zhang. Latent-variable advantage-weighted policy optimization for offline rl. *arXiv preprint arXiv:2203.08949*, 2022.

[CHM$^+$18a] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience, 2018.

[CHM$^+$18b] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan D. Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. *CoRR*, abs/1810.05687, 2018.

[CMH20] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual kitti 2, 2020.

[CPO$^+$19] Wojciech Marian Czarnecki, Razvan Pascanu, Simon Osindero, Siddhant M Jayakumar, Grzegorz Swirszcz, and Max Jaderberg. Distilling policy distillation. *arXiv preprint arXiv:1902.02186*, 2019.

[CPPH$^+$19] Krzysztof Choromanski, Aldo Pacchiano, Jack Parker-Holder, Jasmine Hsu, Atil Iscen, Deepali Jain, and Vikas Sindhwani. When random search is not enough: Sample-efficient and noise-robust blackbox optimization of rl policies. 2019.

[CSAS17] Richard Y Chen, Szymon Sidor, Pieter Abbeel, and John Schulman. Ucb exploration via q-ensembles. *arXiv preprint arXiv:1706.01502*, 2017.

[CSB$^+$17] Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268, 2017.

[CW18] Kamil Ciosek and Shimon Whiteson. Expected Policy Gradients. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second*

*AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 2868–2875. AAAI Press, 2018.

[CZS17] Casey Chu, Andrey Zhmoginov, and Mark Sandler. Cyclegan, a master of steganography. *arXiv preprint arXiv:1712.02950*, 2017.

[CZW$^+$19] Xinyue Chen, Zijian Zhou, Zheng Wang, Che Wang, Yanqiu Wu, and Keith Ross. Bail: Best-action imitation learning for batch deep reinforcement learning, 2019.

[dBKMR] Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross-entropy method.

[DBSV22] Paul Daoudi, Merwan Barlier, Ludovic Dos Santos, and Aladin Virmaux. Density estimation for conservative q-learning, 2022.

[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.

[dHJL19] Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. In *Advances in Neural Information Processing Systems*, pages 11693–11704, 2019.

[DHK$^+$17] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Openai baselines. https://github.com/openai/baselines, 2017.

[DHLDVU16] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Learning and policy search in stochastic dynamical systems with bayesian neural networks. *arXiv preprint arXiv:1605.07127*, 2016.

[DKF20] Jeevan Devaranjan, Amlan Kar, and Sanja Fidler. Meta-sim2: Unsupervised learning of scene structure for synthetic data generation, 2020.

[Dor94] Brigitte Dorner. *Chasing the colour glove: Visual hand tracking*. PhD thesis, Theses (School of Computing Science)/Simon Fraser University, 1994.

[DSC$^+$16] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl\$ˆ2\$: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[DTB$^+$19] Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. Sharing knowledge in multi-task deep reinforcement learning. In *International Conference on Learning Representations*, 2019.

[DWD+21] Yuqing Du, Olivia Watkins, Trevor Darrell, Pieter Abbeel, and Deepak Pathak. Auto-tuned sim-to-real transfer. *CoRR*, abs/2104.07662, 2021.

[DZML12] Guanglong Du, Ping Zhang, Jianhua Mai, and Zeling Li. Markerless kinect-based hand tracking for robot teleoperation. *International Journal of Advanced Robotic Systems*, 9(2):36, 2012.

[EMF20] Clemens Eppner, Arsalan Mousavian, and Dieter Fox. ACRONYM: A large-scale grasp dataset based on simulation. In *2021 IEEE Int. Conf. on Robotics and Automation, ICRA*, 2020.

[ESM+18a] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

[ESM+18b] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[ET94] Bradley Efron and Robert J. Tibshirani. An Introduction to the Bootstrap. *SIAM Review*, 36(4):677–678, 1994.

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017.

[FCSS19] Rasool Fakoor, Pratik Chaudhari, Stefano Soatto, and Alexander J Smola. Meta-q-learning. *arXiv preprint arXiv:1910.00125*, 2019.

[FG21] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *arXiv preprint arXiv:2106.06860*, 2021.

[FHM18] Scott Fujimoto, Herke Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *International Conference on Machine Learning*, pages 1582–1591, 2018.

[FKN+20a] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning. In *arXiv*, 2020.

[FKN+20b] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[FMG22] Hiroki Furuta, Yutaka Matsuo, and Shixiang Shane Gu. Generalized decision transformer for offline hindsight information matching. In *International Conference on Learning Representations*, 2022.

[FMP19] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[FRF$^+$18] Aleksandra Faust, Oscar Ramirez, Marek Fiser, Ken Oslund, Anthony Francis, James Davidson, and Lydia Tapia. Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. pages 5113–5120, Brisbane, Australia, 2018.

[Fro] From MathWorld–A Wolfram Web Resource. *Oblique Coordinates*.

[FSL$^+$17] Bin Fang, Fuchun Sun, Huaping Liu, Di Guo, Wendan Chen, and Guodong Yao. Robotic teleoperation systems using a wearable multimodal fusion device. *International journal of advanced robotic systems*, 14(4):1729881417717057, 2017.

[FvHM18] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *CoRR*, abs/1802.09477, 2018.

[FZG$^+$20] Kuan Fang, Yuke Zhu, Animesh Garg, Andrey Kurenkov, Viraj Mehta, Li Fei-Fei, and Silvio Savarese. Learning task-oriented grasping for tool manipulation from simulated self-supervision. *The International Journal of Robotics Research*, 39(2-3):202–216, 2020.

[GE07] Mohammad Ghavamzadeh and Yaakov Engel. Bayesian actor-critic algorithms. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, pages 297–304, 2007.

[GEV16] Mohammad Ghavamzadeh, Yaakov Engel, and Michal Valko. Bayesian policy gradient and actor-critic algorithms. *Journal of Machine Learning Research*, 17:66:1–66:53, 2016.

[GHLL16] Shixiang Gu, Ethan Holly, Timothy P. Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation. *CoRR*, abs/1610.00633, 2016.

[GLG$^+$17] Shixiang Gu, Timothy P. Lillicrap, Zoubin Ghahramani, Richard E. Turner, and Sergey Levine. Q-Prop: Sample-Efficient Policy Gradient with An Off-Policy Critic. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[GLT$^+$17] Shixiang Gu, Tim Lillicrap, Richard E. Turner, Zoubin Ghahramani, Bernhard Schölkopf, and Sergey Levine. Interpolated Policy Gradient: Merging On-Policy and Off-Policy Gradient Estimation for Deep Reinforcement Learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural*

*Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 3849–3858, 2017.

[GMPT15] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 8(5-6):359–483, 2015.

[GMR16] Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. Improving pilco with bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop, ICML*, volume 4, 2016.

[GPAM+14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Neural Information Processing Systems (NIPS)*, 2014.

[GSR+17] Dibya Ghosh, Avi Singh, Aravind Rajeswaran, Vikash Kumar, and Sergey Levine. Divide-and-conquer reinforcement learning. *arXiv preprint arXiv:1711.09874*, 2017.

[GWCV16] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis, 2016.

[Has10] Hado V. Hasselt. Double Q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621, 2010.

[HBL17] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[HCI+18] Rein Houthooft, Richard Y. Chen, Phillip Isola, Bradly C. Stadie, Filip Wolski, Jonathan Ho, and Pieter Abbeel. Evolved policy gradients. *CoRR*, abs/1802.04821, 2018.

[HCS+17] Karol Hausman, Yevgen Chebotar, Stefan Schaal, Gaurav S. Sukhatme, and Joseph J. Lim. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *NeurIPS*, pages 1235–1245, 2017.

[HE16] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *CoRR*, abs/1606.03476, 2016.

[HGH+19] Jan Humplik, Alexandre Galashov, Leonard Hasenclever, Pedro A Ortega, Yee Whye Teh, and Nicolas Heess. Meta reinforcement learning as task inference. *arXiv preprint arXiv:1905.06424*, 2019.

[HLD$^+$19] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *CoRR*, abs/1901.08652, 2019.

[HRX$^+$20] Daniel Ho, Kanishka Rao, Zhuo Xu, Eric Jang, Mohi Khansari, and Yunfei Bai. Retinagan: An object-aware approach to sim-to-real transfer. *CoRR*, abs/2011.03148, 2020.

[HSE$^+$19] Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3796–3803, 2019.

[HVWY$^+$20] Ankur Handa, Karl Van Wyk, Wei Yang, Jacky Liang, Yu-Wei Chao, Qian Wan, Stan Birchfield, Nathan Ratliff, and Dieter Fox. Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9164–9170. IEEE, 2020.

[HWMD14] Ankur Handa, Thomas Whelan, John McDonald, and Andrew J Davison. A benchmark for rgb-d visual odometry, 3d reconstruction and slam. *ICRA*, 2014.

[HWS$^+$15] Nicolas Heess, Gregory Wayne, David Silver, Timothy P. Lillicrap, Tom Erez, and Yuval Tassa. Learning Continuous Control Policies by Stochastic Value Gradients. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2944–2952, 2015.

[HZAL18a] 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.

[HZAL18b] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning*, pages 1856–1865, 2018.

[HZH$^+$18] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic Algorithms and Applications. *CoRR*, abs/1812.05905, 2018.

[HZJ$^+$21] Muzhi Han, Zeyu Zhang, Ziyuan Jiao, Xu Xie, Yixin Zhu, Song-Chun Zhu, and Hangxin Liu. Reconstructing interactive 3d scenes by panoptic mapping and cad model alignments, 2021.

[ICT$^+$18] Atil Iscen, Ken Caluwaerts, Jie Tan, Tingnan Zhang, Erwin Coumans, Vikas Sindhwani, and Vincent Vanhoucke. Policies modulating trajectory generators. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 916–926. PMLR, 29–31 Oct 2018.

[IKK$^+$22] Maximilian Igl, Daewoo Kim, Alex Kuefler, Paul Mougin, Punit Shah, Kyriacos Shiarlis, Dragomir Anguelov, Mark Palatucci, Brandyn White, and Shimon Whiteson. Symphony: Learning realistic and diverse agents for autonomous driving simulation, 2022.

[JABJ18] Chi Jin, Zeyuan Allen-Zhu, Sébastien Bubeck, and Michael I. Jordan. Is Q-Learning Provably Efficient? In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 4868–4878, 2018.

[JF09] Roland S Johansson and J Randall Flanagan. Coding and use of tactile signals from the fingertips in object manipulation tasks. *Nature Reviews Neuroscience*, 10(5):345–359, 2009.

[JGS$^+$19] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

[JHZ22] Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building digital twins of articulated objects from interaction, 2022.

[JKP21] Se-Yun Jeon, Eun-Su Kim, and Bum Yong Park. Cnn-based hand gesture recognition method for teleoperation control of industrial robot. *IEMEK Journal of Embedded Systems and Applications*, 16(2):65–72, 2021.

[JLK22] Youngsoo Jang, Jongmin Lee, and Kee-Eung Kim. GPT-critic: Offline reinforcement learning for end-to-end task-oriented dialogue systems. In *International Conference on Learning Representations*, 2022.

[JWK$^+$18] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. *CoRR*, abs/1812.07252, 2018.

[Kak01] Sham Kakade. A Natural Policy Gradient. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Syn-*

*thetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pages 1531–1538. MIT Press, 2001.

[Kak02] Sham M Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

[Kak03] Sham M. Kakade. On the sample complexity of reinforcement learning. Phd thesis, University College London, 2003.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[KFS$^+$19a] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pages 11761–11771, 2019.

[KFS$^+$19b] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[KH01] Maria Kozhevnikov and Mary Hegarty. A dissociation between object manipulation spatial ability and spatial orientation ability. *Memory & cognition*, 29(5):745–756, 2001.

[KHSL22] Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. When should we prefer offline reinforcement learning over behavioral cloning?, 2022.

[KIP$^+$18] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *CoRR*, abs/1806.10293, 2018.

[KL02] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning (ICML)*, volume 2, 2002.

[KMH$^+$17] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai, 2017.

[KNL21] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[KPL$^+$19] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets, 2019.

[KSJ$^+$16] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

[KT15] Vikash Kumar and Emanuel Todorov. Mujoco haptix: A virtual reality system for hand manipulation. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 657–663. IEEE, 2015.

[KTFN21] Ilya Kostrikov, Jonathan Tompson, Rob Fergus, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. *arXiv preprint arXiv:2103.08050*, 2021.

[Kum19] Aviral Kumar. Data-driven deep reinforcement learning. https://bair.berkeley.edu/blog/2019/12/05/bear/, 2019. BAIR Blog.

[KVC$^+$21] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *CoRR*, abs/2104.08212, 2021.

[KVN$^+$19] Kirthevasan Kandasamy, Karun Raju Vysyaraju, Willie Neiswanger, Biswajit Paria, Christopher R. Collins, Jeff Schneider, Barnabas Poczos, and Eric P. Xing. Tuning hyperparameters without grad students: Scalable and robust bayesian optimisation with dragonfly. *CoRR*, 2019.

[KvSS19] Louis Kirsch, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Improving generalization in meta reinforcement learning using learned objectives. *arXiv preprint arXiv:1910.04098*, 2019.

[KVW07] Jonathan Kofman, Siddharth Verma, and Xianghai Wu. Robot-manipulator teleoperation by markerless vision-based hand-arm tracking. *International Journal of Optomechatronics*, 1(3):331–357, 2007.

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

[KŽ05] Ladislav Kavan and Jiří Žára. Spherical blend skinning: a real-time deformation of articulated models. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 9–16, 2005.

[KZTL20] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

[Lev17] Sergey Levine. UC Berkeley CS294 deep reinforcement learning lecture notes. http://rail.eecs.berkeley.edu/deeprlcourse-fa17/index.html, 2017.

[LGH$^+$18] Jeongseok Lee1, Michael X. Grey, Sehoon Ha, Tobias Kunz, Sumit Jain, Yuting Ye, Siddhartha S. Srinivasa1, Mike Stilman, , and C. Karen Liu. Dart: Dynamic animation and robotics toolkit. 2018.

[LGR12] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[LH16] Ilya Loshchilov and Frank Hutter. Cma-es for hyperparameter optimization of deep neural networks. *CoRR*, abs/1604.07269, 2016.

[LHC$^+$21] Vincent Lim, Huang Huang, Lawrence Yunliang Chen, Jonathan Wang, Jeffrey Ichnowski, Daniel Seita, Michael Laskey, and Ken Goldberg. Planar robot casting with real2sim2real self-supervised learning, 2021.

[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.

[LHP$^+$16] 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. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[LHW$^+$20] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *CoRR*, abs/2010.11251, 2020.

[LJL$^+$18] Sungsu Lim, Ajin Joseph, Lei Le, Yangchen Pan, and Martha White. Actor-expert: A framework for using action-value methods in continuous action spaces. *CoRR*, abs/1810.09103, 2018.

[LJX$^+$21] Shaowei Liu, Hanwen Jiang, Jiarui Xu, Sifei Liu, and Xiaolong Wang. Semi-supervised 3d hand-object poses estimation with interactions in time. *arXiv preprint arXiv:2106.05266*, 2021.

[LKTF20] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[LLGW19] Lin Lan, Zhenguo Li, Xiaohong Guan, and Pinghui Wang. Meta reinforcement learning with task embedding and shared policy. *arXiv preprint arXiv:1905.06527*, 2019.

[LLLY19] Jing Luo, Zhidong Lin, Yanan Li, and Chenguang Yang. A teleoperation framework for mobile robots based on shared control. *IEEE Robotics and Automation Letters*, 5(2):377–384, 2019.

[LML$^+$19] Shuang Li, Xiaojian Ma, Hongzhuo Liang, Michael Görner, Philipp Ruppel, Bin Fang, Fuchun Sun, and Jianwei Zhang. Vision-based teleoperation of shadow dexterous hand using end-to-end deep neural network. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 416–422. IEEE, 2019.

[LMR$^+$15] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015.

[LSE17] Yunzhu Li, Jiaming Song, and Stefano Ermon. Infogail: Interpretable imitation learning from visual demonstrations. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *NeurIPS*, pages 3812–3822, 2017.

[LTLL22] Liu Liu, Ziyang Tang, Lanqing Li, and Dijun Luo. Robust imitation learning from corrupted demonstrations, 2022.

[LXMM$^+$21] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, Andrey Kurenkov, C. Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks, 2021.

[LZF$^+$21] Zhihan Liu, Yufeng Zhang, Zuyue Fu, Zhuoran Yang, and Zhaoran Wang. Provably efficient generative adversarial imitation learning for online and offline setting with linear function approximation. *CoRR*, abs/2108.08765, 2021.

[Mam16] Khaled Mamou. Volumetric approximate convex decomposition. In Eric Lengyel, editor, *Game Engine Gems 3*, chapter 12, pages 141–158. A K Peters / CRC Press, 2016.

[MCM$^+$17] Chiara Meneghetti, Ramona Cardillo, Irene C Mammarella, Sara Caviola, and Erika Borella. The role of practice and strategy in mental rotation training: transfer and maintenance effects. *Psychological research*, 81(2):415–431, 2017.

[MFB20] Hossein Mobahi, Mehrdad Farajtabar, and Peter L. Bartlett. Self-distillation amplifies regularization in hilbert space, 2020.

[MG21] Liu Minghua and Jiayuan Gu. Mplib. https://github.com/haosulab/MPlib, 2021.

[MGR18] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning. *CoRR*, abs/1803.07055, 2018.

[MGY$^+$21] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim M. Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, Jiwoo Pak, Andy Tong, Kavya Srinivasa, Will Hang, Emre Tuncer, Quoc V.

Le, James Laudon, Richard Ho, Roger Carpenter, and Jeff Dean. A graph placement methodology for fast chip design. *Nature*, 594 7862:207–212, 2021.

[MJKM04] Nathan Miller, Odest Chadwicke Jenkins, Marcelo Kallmann, and Maja J Mataric. Motion capture from inertial sensing for untethered humanoid teleoperation. In *4th IEEE/RAS International Conference on Humanoid Robots, 2004.*, volume 2, pages 547–565. IEEE, 2004.

[MKS$^+$15a] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[MKS$^+$15b] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[MKV$^+$18] A. Rupam Mahmood, Dmytro Korenkevych, Gautham Vasan, William Ma, and James Bergstra. Benchmarking reinforcement learning algorithms on real-world robots. *CoRR*, abs/1809.07731, 2018.

[MLN$^+$17] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *CoRR*, abs/1703.09312, 2017.

[MML$^+$17] Jeffrey Mahler, Matthew Matl, Xinyu Liu, Albert Li, David V. Gealy, and Ken Goldberg. Dex-net 3.0: Computing robust robot suction grasp targets in point clouds using a new analytic model and deep learning. *CoRR*, abs/1709.06670, 2017.

[MON$^+$18] Lars M. Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. *CoRR*, abs/1812.03828, 2018.

[MPH$^+$16] Jeffrey Mahler, Florian T. Pokorny, Brian Hou, Melrose Roderick, Michael Laskey, Mathieu Aubry, Kai Kohlhoff, Torsten Kröger, James Kuffner, and Ken Goldberg. Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1957–1964, 2016.

[MSPK13] Farhan Mohammad, Kishore Reddy Sudini, Varun Puligilla, and Prabhakara Rao Kapula. Tele-operation of robot using gestures. In *2013 7th Asia Modelling Symposium*, pages 67–71, 2013.

[MST$^+$20] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *CoRR*, abs/2003.08934, 2020.

[MSZ94] Richard M. Murray, S. Shankar Sastry, and Li Zexiang. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., USA, 1st edition, 1994.

[MWY$^+$22] Linghui Meng, Muning Wen, Yaodong Yang, chenyang le, Xi yun Li, Haifeng Zhang, Ying Wen, Weinan Zhang, Jun Wang, and Bo XU. Offline pre-trained multi-agent decision transformer, 2022.

[NAS18] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018.

[OAB$^+$18] OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub W. Pachocki, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *CoRR*, abs/1808.00177, 2018.

[OAC18] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 8626–8638, 2018.

[OBPR16] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep Exploration via Bootstrapped DQN. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4026–4034, 2016.

[OOMM18] Brendan O'Donoghue, Ian Osband, Rémi Munos, and Volodymyr Mnih. The uncertainty bellman equation and exploration. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 3836–3845, 2018.

[OTM21] Yoojin Oh, Marc Toussaint, and Jim Mainprice. A system for traded control teleoperation of manipulation tasks using intent prediction from hand gestures. *CoRR*, abs/2107.01829, 2021.

[P$^+$09] Judea Pearl et al. Causal inference in statistics: An overview. *Statistics surveys*, 3:96–146, 2009.

[PAZA17] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *CoRR*, abs/1710.06537, 2017.

[PBB$^+$18] Aayush Prakash, Shaad Boochoon, Mark Brophy, David Acuna, Eric Cameracci, Gavriel State, Omer Shapira, and Stan Birchfield. Structured domain randomization: Bridging the reality gap by context-aware synthetic data. *CoRR*, abs/1810.10093, 2018.

[PBS15] Emilio Parisotto, Jimmy Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *CoRR*, abs/1511.06342, 2015.

[PCZ$^+$19] Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. MCP: learning composable hierarchical control with multiplicative compositional policies. *CoRR*, abs/1905.09808, 2019.

[Pea09] Judea Pearl. *Causality*. Cambridge University Press, 2009.

[Pea10] Judea Pearl. An introduction to causal inference. *The international journal of biostatistics*, 6(2), 2010.

[PGM$^+$19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. cite arxiv:1912.01703Comment: 12 pages, 3 figures, NeurIPS 2019.

[PJS17] J. Peters, D. Janzing, and B. Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT Press, Cambridge, MA, USA, 2017.

[PKZL19] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[PMA10] J. Peters, K. Mülling, and Y. Altün. Relative entropy policy search. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2010.

[PNM$^+$20] Songyou Peng, Michael Niemeyer, Lars M. Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. *CoRR*, abs/2003.04618, 2020.

[PPM17] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

[PS06] Jan Peters and Stefan Schaal. Policy Gradient Methods for Robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2006, October 9-15, 2006, Beijing, China*, pages 2219–2225. IEEE, 2006.

[PS08a] Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

[PS08b] Jan Peters and Stefan Schaal. Natural Actor-Critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

[PS08c] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

[Put14] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

[QCG$^+$09] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[RCG$^+$15] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

[RLTK17a] Aravind Rajeswaran, Kendall Lowrey, Emanuel Todorov, and Sham Kakade. Towards generalization and simplicity in continuous control. *CoRR*, abs/1703.02660, 2017.

[RLTK17b] Aravind Rajeswaran, Kendall Lowrey, Emanuel Todorov, and Sham Kakade. Towards generalization and simplicity in continuous control. *CoRR*, abs/1703.02660, 2017.

[RMS$^+$20] Karsten Roth, Timo Milbich, Samarth Sinha, Prateek Gupta, Bjoern Ommer, and Joseph Paul Cohen. Revisiting training strategies and generalization performance in deep metric learning. *arXiv preprint arXiv:2002.08473*, 2020.

[RMW14] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286. PMLR, 22–24 Jun 2014.

[RRK$^+$21] Rajarshi Roy, Jonathan Raiman, Neel Kant, Ilyas Elkin, Robert Kirby, Michael Siu, Stuart Oberman, Saad Godil, and Bryan Catanzaro. PrefixRL: Optimization of parallel prefix circuits using deep reinforcement learning. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, dec 2021.

[RSC18] Nataniel Ruiz, Samuel Schulter, and Manmohan Chandraker. Learning to simulate. *CoRR*, abs/1810.02513, 2018.

[RSJ20] Yu Rong, Takaaki Shiratori, and Hanbyul Joo. Frankmocap: Fast monocular 3d hand and body motion capture by regression and integration. *arXiv preprint arXiv:2008.08324*, 2020.

[RTB17] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics (ToG)*, 36(6):1–17, 2017.

[RZQ$^+$19] Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. *arXiv preprint arXiv:1903.08254*, 2019.

[SAL$^+$20] Archit Sharma, Michael Ahn, Sergey Levine, Vikash Kumar, Karol Hausman, and Shixiang Gu. Emergent real-world robotic skills via unsupervised off-policy reinforcement learning, 2020.

[SB18] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[SCU$^+$21] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat, 2021.

[SGJ$^+$12] Weibo Song, Xianjiu Guo, Fengjiao Jiang, Song Yang, Guoxing Jiang, and Yunfeng Shi. Teleoperation humanoid robot control system based on kinect sensor. In *2012 4th International Conference on Intelligent Human-Machine Systems and Cybernetics*, volume 2, pages 264–267, 2012.

[SGT$^+$08] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[SHD18] Steindór Sæmundsson, Katja Hofmann, and Marc Peter Deisenroth. Meta reinforcement learning with latent variable gaussian processes. *arXiv preprint arXiv:1803.07551*, 2018.

[SK07] Bruno Siciliano and Oussama Khatib. *Springer Handbook of Robotics*. Springer-Verlag, Berlin, Heidelberg, 2007.

[SKR$^+$15] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, et al. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, pages 3633–3642, 2015.

[SLH$^+$14] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin A. Riedmiller. Deterministic Policy Gradient Algorithms. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 387–395. JMLR.org, 2014.

[SLM$^+$15] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015.

[SML$^+$15] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *CoRR*, abs/1506.02438, 2015.

[SMSM99] Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller, editors, *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, pages 1057–1063. The MIT Press, 1999.

[SMTF21] Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. *CoRR*, abs/2103.14127, 2021.

[SSB$^+$20] Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

[Sut95] Richard S. Sutton. Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding. In David S. Touretzky, Michael Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8, NIPS, Denver, CO, USA, November 27-30, 1995*, pages 1038–1044. MIT Press, 1995.

[SWD10] Debi Stransky, Laurie M Wilcox, and Adam Dubrowski. Mental rotation: cross-task training and generalization. *Journal of Experimental Psychology: Applied*, 16(4):349, 2010.

[SWD$^+$17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347, 2017.

[SYF$^+$22] Charlie Snell, Mengjiao Yang, Justin Fu, Yi Su, and Sergey Levine. Context-aware language modeling for goal-oriented dialogue systems, 2022.

[TAH$^+$04] Christian Theobalt, Irene Albrecht, Jörg Haber, Marcus Magnor, and Hans-Peter Seidel. Pitching a baseball: tracking high-speed motion with multi-exposure images. In *SIGGRAPH 2004 Papers*, pages 540–547. ACM, 2004.

[TAS18] Voot Tangkaratt, Abbas Abdolmaleki, and Masashi Sugiyama. Guide actor-critic for continuous control. In *International Conference on Learning Representations*, 2018.

[TBC$^+$17] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4496–4506, 2017.

[Tes95] Gerald Tesauro. Temporal difference learning and td-gammon. *Commun. ACM*, 38(3):58–68, mar 1995.

[TET12] 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.

[TFR$^+$17] Joshua Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *CoRR*, abs/1703.06907, 2017.

[TtDDT] Russ Tedrake and the Drake Development Team. Notation basics. https://drake.mit.edu/doxygen_cxx/group__multibody__notation__basics.html.

[TtDDT19] Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019.

[TUR50] A. M. TURING. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX(236):433–460, 10 1950.

[TZC$^+$18] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *CoRR*, abs/1804.10332, 2018.

[UHC$^+$19] Yusuke Urakami, Alec Hodgkinson, Casey Carlin, Randall Leu, Luca Rigazio, and Pieter Abbeel. Doorgym: A scalable door opening environment and baseline agent. *arXiv preprint arXiv:1908.01887*, 2019.

[vHDS$^+$18] Hado van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep Reinforcement Learning and the Deadly Triad. *CoRR*, abs/1812.02648, 2018.

[vHGS16] Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-Learning. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2094–2100. AAAI Press, 2016.

[VJK12] Goran Vasiljevic, Nikola Jagodin, and Zdenko Kovacic. Kinect-based robot teleoperation by velocities control in the joint/cartesian frames. *IFAC Proceedings Volumes*, 45(22):805–810, 2012. 10th IFAC Symposium on Robot Control.

[VLL$^+$19] Quan Vuong, Shuang Liu, Minghua Liu, Kamil Ciosek, Hao Su, and Henrik Iskov Christensen. Multi-task batch reinforcement learning with metric learning. *CoRR*, abs/1909.11373, 2019.

[vSvHWW09] Harm van Seijen, Hado van Hasselt, Shimon Whiteson, and Marco A. Wiering. A theoretical and empirical analysis of Expected Sarsa. In *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, ADPRL 2009, Nashville, TN, USA, March 31 - April 1, 2009*, pages 177–184. IEEE, 2009.

[VZR18] Quan Ho Vuong, Yiming Zhang, and Keith W. Ross. Supervised policy update. *CoRR*, abs/1805.11706, 2018.

[WBH$^+$16] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Rémi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *CoRR*, abs/1611.01224, 2016.

[WD92] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[Wil92a] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer, 1992.

[Wil92b] Ronald J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8:229–256, 1992.

[WKT$^+$16] Jane X. Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z. Leibo, Rémi Munos, Charles Blundell, Dharshan Kumaran, and Matthew Botvinick. Learning to reinforcement learn. *CoRR*, abs/1611.05763, 2016.

[WMG$^+$17] Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pages 5285–5294, 2017.

[WNŻ+20] Ziyu Wang, Alexander Novikov, Konrad Żołna, Jost Tobias Springenberg, Scott Reed, Bobak Shahriari, Noah Siegel, Josh Merel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. *arXiv preprint arXiv:2006.15134*, 2020.

[WP09] Robert Y Wang and Jovan Popović. Real-time hand-tracking with a color glove. *ACM transactions on graphics (TOG)*, 28(3):1–8, 2009.

[WPC+20] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[WTN19] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[WWVR20] Che Wang, Yanqiu Wu, Quan Vuong, and Keith Ross. Striving for simplicity and performance in off-policy drl: Output normalization and non-uniform sampling, 2020.

[XCB+22] Zhenjia Xu, Cheng Chi, Benjamin Burchfiel, Eric Cousineau, Siyuan Feng, and Shuran Song. Dextairity: Deformable manipulation can be a breeze, 2022.

[XLHL19] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. Self-training with noisy student improves imagenet classification, 2019.

[XLZ+19] Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? *arXiv preprint arXiv:1905.13211*, 2019.

[XQM+20] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11097–11107, 2020.

[YKG+19] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Multi-task reinforcement learning without interference. 2019.

[YKTL19] Wenhao Yu, Visak CV Kumar, Greg Turk, and C. Karen Liu. Sim-to-real transfer for biped locomotion. 2019.

[YLT18] Wenhao Yu, C. Karen Liu, and Greg Turk. Policy transfer with strategy optimization. *CoRR*, abs/1810.05751, 2018.

[YXWW20] Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. *arXiv preprint arXiv:2003.13661*, 2020.

[ZBH20] Wenxuan Zhou, Sujay Bajracharya, and David Held. Plas: Latent action space for offline reinforcement learning. *arXiv preprint arXiv:2011.07213*, 2020.

[ZBV$^+$20] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*, 2020.

[ZCZ$^+$18] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

[ZGB$^+$17] M. Zhang, X. Geng, J. Bruce, K. Caluwaerts, M. Vespignani, V. SunSpiral, P. Abbeel, and S. Levine. Deep reinforcement learning for tensegrity robot locomotion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 634–641, May 2017.

[Zie10] Brian D. Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD Thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2010.

[ZPK18] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing, 2018. cite arxiv:1801.09847Comment: http://www.open3d.org.

[ZSI$^+$20] Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. In *International Conference on Learning Representation (ICLR)*, 2020.

[ZSK$^+$19] Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7693–7702, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[ZTC$^+$20] Ilya Zakharkin, Arman Tsaturyan, Miguel Altamirano Cabrera, Jonathan Tirado, and Dzmitry Tsetserukou. Zoomtouch: Multi-user remote robot control in zoom by dnn-based gesture recognition. In *SIGGRAPH Asia 2020 Emerging Technologies*, SA '20, New York, NY, USA, 2020. Association for Computing Machinery.

[ZVS$^+$18] Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew J. Johnson, and Sergey Levine. SOLAR: deep structured latent representations for model-based reinforcement learning. *CoRR*, abs/1808.09105, 2018.