Towards Adaptive, Continual Embodied Agents

by

Kelvin Xu

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Summer 2022

Towards Adaptive, Continual Embodied Agents

Abstract

Towards Adaptive, Continual Embodied Agents

by

Kelvin Xu

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Sergey Levine, Chair

In recent years, artificial learning systems have demonstrated tremendous advances on a number of challenging domains such as computer vision, natural language processing and speech recognition. A striking characteristic of these recent advances has been the seemingly simple formula of combining flexible deep function approximators with large datasets collected for specific problems. These systems struggle however to leverage their learned capabilities when generalizing to new inputs for acquiring new capabilities, often requiring re-training from scratch on a similarly large dataset from scratch. This is in stark contrast to humans, who have a remarkable ability to build upon their prior experiences and learn new concepts from only a few examples. In the first part of this thesis, we will study the question of how to construct systems that mimic this ability to adapt rapidly to new tasks. One of the core principles that will underlie this part of the thesis will be to leverage structure in a large number of prior experiences/tasks to enable fast adaptation and uncertainty. We will start first by studying the setting of reward specification, a common challenge in reinforcement learning, and next study how a probabilistic framing of the meta-learning setting can enable reasoning under uncertainty.

In the second part of this thesis, given the established potential that a prior datasets of tasks can play in accelerating learning, we will ask the natural question of how to enable agents to collect data completely autonomously. This would remove the need of a human to “curate” the dataset of tasks for the artificial agent and enable fully scalable never ending embodied learning. The central theme of the approach we take will be to consider the online real world nature of “tasks” that an agent must solve, and through it revisit the basic assumptions of episodic RL. Finally, we will conclude with a demonstration of these ideas in the domain of real world dexterous manipulation and provide some hints for future work in this more “autonomous” reinforcement learning setting.

ACKNOWLEDGMENTS

Throughout my PhD, I have been very fortunate to cross paths with a special group of people. This journey would not have been possible for their support and encouragement.

First and foremost, thank you Sergey Levine for being a great mentor and role model. It has been a privilege to be a part of your group. Your intellect, dedication to science and ability to navigate through thick and thin have served as an inspiration for me through my time here. Thank you to my committee members Pieter Abbeel, Alyosha Efros and Alison Gopnik for your support and guidance through this process. Thank you Yoshua Bengio, Aaron Courville, Kyunghyun Cho, Jamie Kiros and Aaron Judah for words of encouragement at the beginning of my PhD journey.

Throughout this thesis work, I have been fortunate to collaborate with an amazing set of fellow students, faculty and researchers in the field. I would like to thank Ellis Ratner, Justin Yu, Thomas Devlin, Tony Zhao, Nikhil Sardana, Archit Sharma, Karol Hausman, Vikash Kumar and Anca Dragan for all the enjoyable collaborations. I would like to give a special thanks to the undergraduate students I have worked with for their hard work and enthusiasm: Siddharth Verma, Aaron Rovinsky, Zheyuan Hu, and Ria Doshi. If there is anything I can be confident was a good thing that I did during my PhD, it was serving as a mentor during our work together. Thank you Marc Bellemare, for hosting me during a very fun internship at Google and for giving me a crash course in RL and random graph theory. Finally, a special shout out to Chelsea Finn and Abhishek Gupta who, in addition to being amazing collaborators, have been mentors and sources of encouragement at different points during this process. All I can say is that I am happy that you two are part of the next generation of mentors in academia.

I have also been lucky to interact with all the researchers in RAIL, but I would particularly like to thank some of the students I shared the majority of this journey with: Justin Fu, Avi Singh, Vitchyr Pong, Michael Chang, JD Co-Reyes, Ashvin Nair, Marvin Zhang, Young Geng, Aviral Kumar, Michael Janner, Dhruv Shah, Nick Rhinehart, Natasha Jaques, Rowan McAllister, Ilya Kostrikov and many more. Thank you Coline Devine, Kate Rakelly, Alex Lee and Greg Kahn for also being amazing sources of counsel in my final years of grad school. Finally, I'd like to add a note of appreciation to Homer Walke, Dibya Ghosh, Katie Kang, Colin Li and Laura Smith for being a fun group that I could look forward to having lunch with when we returned to being in person.

I would like to also thank the broader BAIR/Berkeley community for providing a very stimulating environment, but mostly for the friends I've made along the way. In particular, I'd like to thank Clara Fannjiang, Parsa Mahmoudieh, Allan Jabri, Nilesh Tripuraneni, Ilija Radosavovic, Ashish Kumar, Toru Lin, Akosua Busia, Thanard Kurutach, Suzie Petryk, Armin Askari, Evonne Ng, Sasha Sax, Utku Evci, Brian Cheung, Erin Grant, Andrew Waterman, Chloe Hsu, Melih Elibol, Vickie Ye, Gabriel Enrique Colón, Wendy Shang, Isabella Huang and many more for all the fun nights out, baby yoda memes, BWW hangs, surprise gifts, bourbon tasting, missed flights in Europe and support at the difficult times in this journey. Lastly, special shout out to Jasmine Collins, without whom this journey would not have been possible, and definitely not as fun. You're a pal.

Thank you to my other friends here, there and everywhere for their support, encouragement, and reminders to enjoy the ride. In particular, I'd like to thank Megan Baker, Amanda Chou, Jay Louie, Sam Edwards, Mika Weissbuch, Jeevan Gyawali, Mia Borzello, Stephan Kaufhold, Orhan Firat, Junyoung Chung, Alexander Fengler, Wendy Shang, Barret Zoph, and Liam Fedus. Finally, a special shout out to the Toronto fam for all the support during ups and downs. To David Biancolin, Fiona Macleod, Richard Gao, Zimu Zhu, Kevin Lam, Judith Ma, Matt Nejati, Denise Punzal, Mohamad Kayed, and many more, a million thanks.

I feel very lucky to have done the work contained in this thesis in a number of beautiful places. I feel grateful to have imprinted in me memories of climbing around California, channel orange surf sessions at Scripps, hikes in firetrails, the Taco Bell in Pacifica, Berkeley Bowl, Ironworks, Lake Merritt, and finally the styling neighborhood of Prenzlauer Berg where I wrote this dissertation. Having places I loved to escape to was immeasurably valuable during this process, especially the pandemic years.

Last, but not least, I would like to thank my parents (Wei Xu and Huijing Chen) for supporting me and giving me a life of opportunities. Thank you to my sister, Hailen Xu, for always being a good sport. Finally, thank you to my partner Vicky Zhao for having patience, offering encouragement when I needed it, and helping me enjoy the journey.

CONTENTS

1 INTRODUCTION

Arguably, one of the central appeals aspects of learning based approaches to embodied systems are that they provide a *naturalistic* account of how complex behavior could emerge autonomously from interaction, quite similar to how humans and animals can acquire skill through experience. Rather than rely on a human to provide all of an agent's knowledge a priori, the agent could itself acquire useful behavior from trial and error. In addition, learning based approaches in principle provide a powerful mechanism for agents to be *adaptive* in the face of change, an inevitable consideration for any system in an open world environment. Consider for example a home robot. We would ideally like the robot to autonomously learn the particularities of the home without human programming by learning good strategies through interaction, and adapt when the home changes. In addition, given changes in the home, we would like our systems to reuse knowledge it has gained before, rather than relearn everything completely from scratch.

This futuristic vision of autonomous, adaptive learning systems are in stark contrast however to the controlled experimental settings typical in the robot learning literature (Levine et al., 2016a; Chebotar et al., 2017; Yahya et al., 2017; Ghadirzadeh et al., 2017b; Zeng et al., 2020). It also contrasts strongly with the dominant paradigm in the problem of supervised learning which is to have a human collect and label a large amount of data to supervise the training of a randomly initialized deep neural network (Krizhevsky et al., 2012; Hinton et al., 2012; Sutskever et al., 2014). From a naturalistic intelligence perspective, this is unsatisfying, as humans are able to learn in very open ended environments and can reuse knowledge when learning new behaviors. It is also a challenge from a practical perspective, as modern ML systems have particularly excelled in settings where large labeled dataset can be collected and deployed in the closed environments. These assumptions do not hold universally however, and often the main bottleneck to applying ML in such settings can be the human effort needed for data collection or environment engineering. Thus, in order for real world learning systems to realize their fullest potential in terms of practicality and scalability, it is critical to design tractable learning algorithms that limit such assumptions.

This thesis takes steps toward building systems that are better equipped to deal with the adaptive, continual nature of the real world. In the first part of this thesis, we will investigate how we can allow agents to leverage prior tasks to both accelerate learning and handle uncertainty when only given limited supervision. We do so by leveraging the meta-learning framework (Schmidhuber, 1987b; *Learning a synaptic learning rule*; Thrun and Pratt, 1998), which can be viewed as learning a learning algorithm via components such as weight initialization or optimizer (Finn et al., 2017a; Ravi and Larochelle, 2017). We argue that meta-learning is a natural fit for problems faced by embodied agents and present an application in reward specification via inverse reinforcement learning. We also present work extending these methods in the face of uncertainty by framing our approach as a probabilistic inference problem. We show how by leveraging prior tasks, we can both learn efficiently from limited supervision using structure in related tasks. We additionally show how the uncertainty measures we derive can be used in principled decision making schemes.

Having established the potential of leveraging previous experience for rapid adaptation, in the second part of this thesis we turn to the practical challenge of how agents can collect data autonomously in a lifelong manner. This ability, combined with the ability to reuse prior capabilities, promises to allow for lifelong agents to continually improve and acquire increasingly complex behavior. In addressing these challenges, we examine the setting of never ending data collection as an extension of the standard episodic RL framework. We study formalisms, algorithms and finally real world applications in this setting. We present in this part of the thesis real world robotic experiments of a dexterous four fingered robot, which learns to successfully manipulate complex objects, learn multiple behaviors, and autonomously recover from failure.

The main contributions of this thesis are organized as follows:

- In Chapter 2, we present an application of meta-learning in the setting of reward specification via inverse reinforcement learning. We build on prior meta-learning approaches and additionally describe an interpretation of our approach as learning a probabilistic 'prior' over intent.
- Next, in Chapter 3, we extend this link to probabilistic inference to allow for a more principled approach to handling uncertainty. We demonstrate the efficacy of this approach in the standard few-shot classification setting, in addition to the active learning setting.
- Having established the potential of leveraging structure in previously experienced

"tasks", in chapter 4, we next shift to the question of how embodied agents can autonomously collect their own set of tasks autonomously. We do so by studying a modification of the standard episodic RL setting and by using ideas from unsupervised skill learning.

- In Chapter 5 and 6, we present the culmination of these ideas in the challenging setting of real world dexterous manipulation. We show how through hundreds of hours of continuous interaction, we can learn multiple complex behaviors and leverage them for continual data collection. In bridging these ideas to the real world, we additionally address challenges in reward inference, state estimation, and task inference in a unified data driven manner.
- In Chapter 7, we present the present state of algorithms in the setting of "Autonomous Reinforcement Learning" by presenting both a benchmark and a comprehensive evaluation. We describe qualitative shortcomings in current approaches, and suggest future directions for research in this setting.
- Finally, in Chapter 8, we conclude with future directions for tackling challenges in learning on embodied systems.

These chapters cover work from the papers listed below. Where a "*" is indicated, the set of authors contributed equally. Where I was the sole first author of the paper, it indicates that I was principally responsible for leading the research and writing of the work. Where multiple authors contributed as first authors, it indicates that the experimental and written contributions were equally shared and the work was co-led. In the work covered in Chapter 5, the work was primarily led by my co-authors of Gupta et al., 2021a, where I contributed to the simulation results. In the event of any uncertainty related to work attribution, all credit should go to my co-authors without whom this thesis would not have been possible.

1. K. Xu, Z. Hu, A. Rovinsky, R. Doshi, A. Gupta, V. Kumar, S. Levine, Autonomous Multitask Visual Dexterous Manipulation, In submission (2022)
2. A. Sharma*, K. Xu*, A. Gupta, K. Hausman, S. Levine, C. Finn, Autonomous Reinforcement Learning: Formalism and Benchmarking, ICLR (2022)
3. A. Gupta*, J. Yu*, T. Zhao*, V. Kumar*, A. Rovinsky, K. Xu, T. Devlin, S. Levine, Reset-Free Reinforcement Learning via Multi-Task Learning: Learning Dexterous Manipulation Behaviors without Human Intervention, ICRA (2021)
4. K. Xu*, S. Verma*, C. Finn, S. Levine, Learning Skillful Resets: Acquisition of Behaviors via Reset-Free Games, NeurIPS (2020)
5. K. Xu, E. Ratner, A. Dragan, S. Levine, C. Finn, Learning a Prior over Intent via Meta-Inverse Reinforcement Learning, ICML (2019)
6. C. Finn*, K. Xu*, S. Levine, Probabilistic Model-Agnostic Meta-Learning, NeurIPS (2018)

2

LEARNING A PRIOR OVER INTENT VIA META-INVERSE REINFORCEMENT LEARNING

Reinforcement learning (RL) algorithms have the potential to automate a wide range of decision-making and control tasks across a variety of different domains, as demonstrated by successful recent applications ranging from robotic control (Kober and Peters, 2012; Levine et al., 2016a) to game playing (Mnih et al., 2015; Silver et al., 2016). A key assumption of the RL problem statement is the availability of a reward function that accurately describes the desired task. For many real world tasks, reward functions can be challenging to manually specify, while being crucial for good performance (Amodei et al., 2016). Most real world tasks are multifaceted and require reasoning over multiple factors in a task (e.g. a robot cleaning in a house with children), while simultaneously providing appropriate reward shaping to make the task feasible with tractable exploration (Ng et al., 1999). These challenges are compounded by the inherent difficulty of specifying rewards for tasks with high-dimensional observation spaces such as images.

Inverse reinforcement learning (IRL) is an approach that aims to address this problem by instead inferring the reward function from demonstrations of the task (Ng and Russell, 2000). This has the appealing benefit of taking a data-driven approach to reward specification in place of hand engineering. In practice however, rewards functions are rarely learned as it can be prohibitively expensive to provide demonstrations that cover the variability common in real world tasks (e.g., collecting demonstrations of opening every type of door knob). In addition, while learning a complex function from high dimensional observations might make an expressive function approximator seem like a reasonable modelling assumption, in the "few-shot" domain it is notoriously difficult to unambiguously recover a good reward function with expressive function approximators. Many prior approaches have thus relied on low-dimensional linear models with hand-crafted features that effectively encode a strong prior on the relevant features of a task. This requires engineering a set of features by hand that work well for a specific problem.

In this work, we propose an approach that instead explicitly learns expressive features that are robust even when learning with limited demonstrations.

Our approach relies on the key observation that related tasks share a common structure that we can leverage when learning new tasks. To illustrate, considering a robot navigating through a home. While the exact reward function we provide to the robot may differ depending on the task, there is a structure amid the space of useful behaviours, such as navigating to a series of landmarks, and there are *certain behaviors* we always want to encourage or discourage, such as avoiding obstacles or staying a reasonable distance from humans. This notion agrees with our understanding of why humans can easily infer the intents and goals (i.e., reward functions) of even abstract agents from just one or a few demonstrations Baker et al., 2007, as humans have access to strong priors about how other humans accomplish similar tasks accrued over many years. Similarly, our objective is to discover the common structure among different tasks, and encode that structure in a way that can be used to infer reward functions from a few demonstrations.

More specifically, in this work we assume access to a set of tasks, along with demonstrations of the desired behaviors for those tasks, which we refer to as the *meta-training set*. From these tasks, we then learn a reward function parameterization that enables effective few-shot learning when used to initialize IRL in a novel task. Our method is summarized in Fig. 1. Our key contribution is an algorithm that enables efficient learning of new reward functions by using meta-training to build a rich "prior" for goal inference. Using our proposed approach, we show that we can learn deep neural network reward functions from raw pixel observations on two distinct domains with substantially better data efficiency than existing methods and standard baselines.

2.1 RELATED WORK

Inverse reinforcement learning (IRL) (Ng and Russell, 2000) is the problem of inferring an expert's reward function directly from demonstrations. Prior methods for performing IRL range from margin based approaches (Abbeel and Ng, 2004; Ratliff et al., 2006a) to probabilistic approaches (Ramachandran and Amir, 2007; Ziebart et al., 2008b). Although it is possible to extend our approach to any other IRL method, in this work we base on work on the maximum entropy (MaxEnt) framework (Ziebart et al., 2008b). In addition to allowing for sub-optimality in the expert demonstrations, MaxEnt-IRL can be re-framed as a maximum likelihood estimation problem. (Sec. 6.3).

In part to combat the under-specified nature of IRL, prior work has often used low-dimensional linear parameterizations with handcrafted features (Abbeel and Ng, 2004; Ziebart et al., 2008b). In order to learn from high dimensional input, Wulfmeier et al.

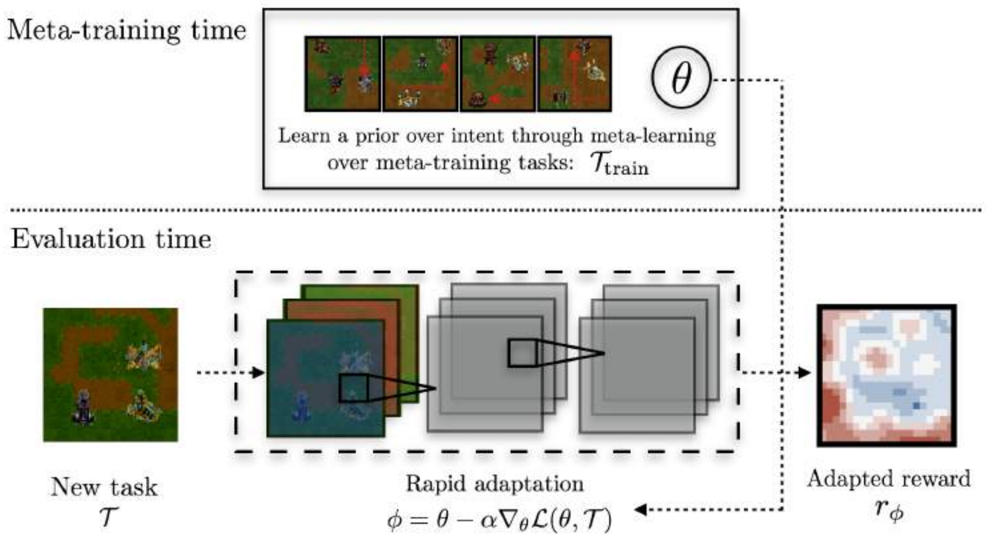

Meta-training time
θ
Learn a prior over intent through meta-learning over meta-training tasks: $\mathcal{T}_{\text{train}}$
Evaluation time
New task $\mathcal{T}$
Rapid adaptation $\phi = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{T})$
Adapted reward r_ϕ

Figure 1: A diagram of our meta-inverse RL approach. Our approach attempts to remedy over-fitting in few-shot IRL by learning a "prior" that constraints the set of possible reward functions to lie within a few steps of gradient descent. Standard IRL attempts to recover the reward function directly from the available demonstrations. The shortcoming of this approach is that there is little reason to expect generalization as it is analogous to training a density model with only a few examples.

(2015a) proposed applying fully convolutional networks (Shelhamer et al., 2017) to the MaxEnt IRL framework (Ziebart et al., 2008b) for several navigation tasks (Wulfmeier et al., 2016a; Wulfmeier et al., 2016b). Other methods that have incorporated neural network rewards include guided cost learning (GCL) (Finn et al., 2017b), which uses importance sampling and regularization for scalability to high-dimensional spaces, and adversarial IRL (Fu et al., 2018a). Several other methods have also proposed imitation learning approaches based on adversarial frameworks that resemble IRL, but do not aim to directly recover a reward function (Ho and Ermon, 2016; Li et al., 2017a; Hausman et al., 2017; Kuefler and Kochenderfer, 2018). In this work, instead of improving the ability to learn reward functions on a single task, we focus on the problem of effectively learning to use prior demonstration data from other IRL tasks, allowing us to learn new tasks from a limited number demonstrations even with expressive non-linear reward functions.

Prior work has explored the problem of *multi-task* IRL, where the demonstrated behavior is assumed to have originated from multiple experts achieving different goals. Some of these approaches include those that aim to incorporate a shared prior over reward functions through extending the Bayesian IRL (Ramachandran and Amir, 2007) framework to the multi-task setting (Dimitrakakis and Rothkopf, 2012; Choi and Kim, 2012). Other approaches have clustered demonstrations while simultaneously inferring reward functions for each cluster (Babeş-Vroman et al., 2011) or introduced regularization between rewards to a common "shared reward" (Li and Burdick, 2017). Our work is similar in that we also seek to encode prior information common to the tasks. However, a critical difference is that our method specifically aims to distill the meta-training tasks into a prior that can then be used to learn rewards for *new* tasks efficiently. The goal therefore is not to acquire good reward functions that explain the meta-training tasks, but rather to use them to learn efficiently on new tasks.

Our approach builds on work on the broader problem of meta-learning (Schmidhuber, 1987b; *Learning a synaptic learning rule*; Naik and Mammone, 1992; Thrun and Pratt, 2012) and generative modelling (Rezende et al., 2016; Reed et al., 2018; Mordatch, 2018). Prior work has proposed a variety of solutions for learning to learn including memory based methods (Duan et al., 2016; Santoro et al., 2016; Wang et al., 2016; Mishra et al., 2017), methods that learn an optimizer and/or initialization (Andrychowicz et al., 2016; Ravi and Larochelle, 2016; Finn et al., 2017b; Li and Malik, 2017), and methods that compare new datapoints in a learned metric space (Koch, 2015; Wang and Hebert, 2016; Vinyals et al., 2016; Shyam et al., 2017; Snell et al., 2017). Our work is motivated by the goal of broadening the applicability of IRL, but in principle it is possible to adapt many of these meta-learning approaches for our problem statement. We build upon Finn et al. (2017b), which has also been previously applied to the related problems of imitation learning and human motion prediction Wang et al., 2016; Finn et al., 2017d; Alet et al., 2018. We leave it to future work to do a comprehensive investigation of different meta-learning approaches which could broaden the applicability of IRL.

2.2 PRELIMINARIES AND OVERVIEW

In this section, we introduce our notation and describe the IRL and meta-learning problems.

2.2.1 *Learning Rewards via Maximum Entropy Inverse Reinforcement Learning*

The standard Markov decision process (MDP) is defined by the tuple $(\mathcal{S}, \mathcal{A}, p_{\mathbf{s}}, R, \gamma)$ where $\mathcal{S}$ and $\mathcal{A}$ denote the set of possible states and actions respectively, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in [0, 1]$ is the discount factor and $p_{\mathbf{s}} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ denotes the transition distribution over the next state $\mathbf{s}_{t+1}$, given the current state $\mathbf{s}_t$ and current action $\mathbf{a}_t$. Typically, the goal of "forward" RL is to maximize the expected discounted return $R(\tau) = \sum_{t=1}^{T} \gamma^{t-1} r(\mathbf{s}_t, \mathbf{a}_t)$.

In IRL, we instead assume that the reward function is unknown but that we instead have access to a set of expert demonstrations $\mathcal{D} = \{\tau_1, \ldots, \tau_K\}$, where $\tau_k = \{\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T\}$.

The goal of IRL is to recover the unknown reward function R from the set of demonstrations. We build on the maximum entropy (MaxEnt) IRL framework by Ziebart et al. (2008b), which models trajectories as being distributed proportional to their exponentiated return

$$p(\tau) = \frac{1}{Z} \exp\left(R(\tau)\right), \tag{1}$$

where Z is the partition function, $Z = \int_\tau \exp(R(\tau)) d\tau$. This distribution can be shown to be induced by the optimal policy in entropy regularized forward RL problem:

$$\pi^* = \arg\max_\pi \mathbb{E}_{\tau \sim \pi} \left[R(\tau) - \log \pi(\tau)\right]. \tag{2}$$

This formulation allows us to pose the reward learning problem as a maximum likelihood estimation (MLE) problem in an energy-based model R_ϕ by defining the following loss:

$$\min_\phi \mathbb{E}_{\tau \sim \mathcal{D}} \left[\mathcal{L}_{\text{IRL}}(\tau)\right] := \min_\phi \mathbb{E}_{\tau \sim \mathcal{D}} \left[-\log p_\phi(\tau)\right]. \tag{3}$$

Learning in general energy-based models of this form is common in many applications such as structured prediction. However, in contrast to applications where learning can be supervised by millions of labels (e.g. semantic segmentation), the learning problem in Eq. 3 must typically be performed with a relatively small number of example demonstrations. In this work, we seek to address this issue in IRL by providing a way to integrate information from prior tasks to constrain the optimization in Eq. 3 in the regime of limited demonstrations.

2.2.2 *Meta-Learning*

The goal of meta-learning algorithms is to optimize for the ability to learn efficiently on new tasks. Rather than attempting to generalize to new datapoints, meta-learning can be understood as attempting to generalize to *new tasks*. It is assumed in the meta-learning setting that there are two *disjoint* sets of tasks that we refer to as the meta-training set $\{\mathcal{T}_i \, ; \, i = 1..N\}$ and meta-test set $\{\mathcal{T}_j \, ; \, j = 1..M\}$, which are both drawn from a distribution $p(\mathcal{T})$. During meta-training time, the meta-learner attempts to learn the structure of the tasks in the meta-training set, such that when it is presented with a test task, it can leverage this structure to learn efficiently from a limited number of examples.

To illustrate this distinction, consider the case of few-shot learning setting. Let f_θ

denote the learner, and let a task be defined by learning from K training examples $X_{\mathcal{T}}^{\text{tr}} = \{\mathbf{x}_1 \ldots, \mathbf{x}_K\}$, $Y_{\mathcal{T}}^{\text{tr}} = \{\mathbf{y}_1 \ldots, \mathbf{y}_K\}$, and evaluating on K' test examples $X_{\mathcal{T}}^{\text{test}} = \{\mathbf{x}_1 \ldots, \mathbf{x}_{K'}\}$, $Y_{\mathcal{T}}^{\text{test}} = \{\mathbf{y}_1 \ldots, \mathbf{y}_{K'}\}$. One approach to meta-learning is to directly parameterize the meta-learner with an expressive model such as a recurrent or recursive neural network (Duan et al., 2016; Mishra et al., 2017) conditioned on the task training data and the inputs for the test task: $f_\theta(Y|X_{\mathcal{T}}^{\text{test}}, X_{\mathcal{T}}^{\text{tr}}, Y_{\mathcal{T}}^{\text{tr}})$. Such a model is optimized using log-likelihood across all tasks. In this approach to meta-learning, since neural networks are known to be universal function approximators (Siegelmann and Sontag, 1995), any desired structure between tasks can be implicitly encoded.

Rather than learn a single black-box function, another approach to meta-learning is to learn components of the learning procedure such as the initialization (Finn et al., 2017b) or the optimization algorithm (Ravi and Larochelle, 2016; Andrychowicz et al., 2016). In this work we extend the approach of model agnostic meta-learning (MAML) introduced by Finn et al. (2017b), which learns an initialization that is adapted by gradient descent. Concretely, in the supervised learning case, given a loss function $\mathcal{L}(\theta, X_{\mathcal{T}}, Y_{\mathcal{T}})$ (e.g. cross-entropy), MAML performs the following optimization

$$\begin{aligned} &\min_\theta \sum_{\mathcal{T}} \mathcal{L}(\boldsymbol{\phi}_{\mathcal{T}}, X_{\mathcal{T}}^{\text{test}}, Y_{\mathcal{T}}^{\text{test}}) \\ =&\min_\theta \sum_{\mathcal{T}} \mathcal{L}\left(\theta - \alpha \nabla_\theta \mathcal{L}(\theta, X_{\mathcal{T}}^{\text{tr}}, Y_{\mathcal{T}}^{\text{tr}}), X_{\mathcal{T}}^{\text{test}}, Y_{\mathcal{T}}^{\text{test}}\right), \end{aligned} \tag{4}$$

where the optimization is over an initial set of parameters θ and the loss on the held out tasks $X_{\mathcal{T}}^{\text{test}}$ becomes the signal for learning the initial parameters for gradient descent (with step size α) on $X_{\mathcal{T}}^{\text{tr}}$. This optimization is analogous to adding a constraint in a multi-task setting, which we show in later sections is analogous in our setting to learning a prior over reward functions.

2.3 LEARNING TO LEARN REWARDS

Our goal in meta-IRL is to learn how to learn reward functions across many tasks such that the model can infer the reward function for a new task using only one or a few expert demonstrations. Intuitively, we can view this problem as aiming to learn a prior over the rewards of expert demonstrators, such that when given just one or a few demonstrations of a new task, we can combine the learned prior with the new data to effectively determine the expert's intent. Such a prior is helpful in inverse reinforcement learning settings, since the space of reward functions with are relevant to particular task is much smaller than the space of all possible rewards definable on the raw observations.

During meta-training, we have a set of tasks $\{\mathcal{T}_i \ ; \ i = 1..N\}$. Each task $\mathcal{T}_i$ has a set of demonstrations $\mathcal{D}_{\mathcal{T}} = \{\tau_1, \ldots, \tau_K\}$ from an expert policy which we partition into disjoint $\mathcal{D}_{\mathcal{T}}^{tr}$ and $\mathcal{D}_{\mathcal{T}}^{test}$ sets. The demonstrations for each meta-training task are assumed to be produced by the expert according to the maximum entropy model in Section 2.2.1. During meta-training, these tasks will be used to encodes common structure so that our model can quickly acquire rewards for new tasks from just a few demonstrations.

After meta-training, our method is presented with a new task. During this meta-test phase, the algorithm must infer the parameters of the reward function $r_\phi(\mathbf{s}_t, \mathbf{a}_t)$ for the new task from a few demonstrations. As is standard in meta-learning, we assume that the test task is from the same distribution of tasks seen during meta-training, a distribution that we denote as $p(\mathcal{T})$.

2.3.1 *Meta Reward and Intention Learning (MandRIL)*

In order to meta-learn a reward function that can act as a prior for new tasks and new environments, we first formalize the notion of a good reward by defining a loss $\mathcal{L}_{\mathcal{T}}(\theta)$ on the reward function r_θ for a particular task $\mathcal{T}$. We use the MaxEnt IRL loss $\mathcal{L}_{\mathrm{IRL}}$ discussed in Section 6.3, which, for a given $\mathcal{D}_{\mathcal{T}}$, leads to the following gradient (Ziebart et al., 2008b):

$$\nabla_\theta \mathcal{L}_{\mathcal{T}}(\theta) = \frac{\partial r_\theta}{\partial \theta} \left[\mathbb{E}_\tau[\boldsymbol{\mu}_\tau] - \boldsymbol{\mu}_{\mathcal{D}_{\mathcal{T}}} \right]. \tag{5}$$

where $\boldsymbol{\mu}_\tau$ are the state-action visitations under the optimal maximum entropy policy under r_θ, and $\boldsymbol{\mu}_{\mathcal{D}}$ are the mean state visitations under the demonstrated trajectories.

If our end goal were to achieve a single reward function that works as well as possible across all tasks in $\{\mathcal{T}_i \ ; \ i = 1..N\}$, then we could simply follow the *mean* gradient across all tasks. However, our objective is different: instead of optimizing performance on the meta-training tasks, we aim to learn a reward function that can be quickly and efficiently adapted to new tasks at meta-test time. In doing so, we aim to encode prior information over the task distribution in this learned reward prior.

We propose to implement such a learning algorithm by finding the parameters θ, such that starting from θ and taking a small number of gradient steps on a few demonstrations from given task leads to a reward function for which a set of *test* demonstrations have high likelihood, with respect to the MaxEnt IRL model. In particular, we would like to find a θ such that the parameters

$$\phi_{\mathcal{T}} = \theta - \alpha \nabla_{\theta} \mathcal{L}^{\text{tr}}_{\mathcal{T}}(\theta) \tag{6}$$

lead to a reward function $r_{\phi_{\mathcal{T}}}$ for task $\mathcal{T}$, such that the IRL loss (corresponding to negative log-likelihood) for a disjoint set of test demonstrations, given by $\mathcal{L}^{\mathcal{T},\text{test}}_{\text{IRL}}$, is minimized. The corresponding optimization problem for θ can therefore be written as follows:

$$\min_{\theta} \sum_{i=1}^{N} \mathcal{L}^{\text{test}}_{\mathcal{T}_i}(\phi_{\mathcal{T}_i}) = \sum_{i=1}^{N} \mathcal{L}^{\text{test}}_{\mathcal{T}_i}\left(\theta - \alpha \nabla_{\theta} \mathcal{L}^{\text{tr}}_{\mathcal{T}_i}(\theta)\right). \tag{7}$$

Our method acquires this prior θ over rewards in the task distribution $p(\mathcal{T})$ by optimizing this loss. This amounts to an extension of the MAML algorithm in Section 2.2.2 to the inverse reinforcement learning setting. This extension is quite challenging, because computing the MaxEnt IRL gradient requires repeatedly solving for the current maximum entropy policy and visitation frequencies, and the MAML objective requires computing derivatives *through* this gradient step. Next, we describe in detail how this is done. An overview of our method is also outlined in Alg. 1.

Meta-training. The computation of the meta-gradient for the objective in Eq. 7 can be conceptually separated into two parts. First, we perform the update in Eq. 6 by computing the *expected state visitations* μ, which is the expected number of times an agent will visit each state. We denote this overall procedure as State-Visitations-Policy, and follow Ziebart et al. (2008b) by first computing the maximum entropy optimal policy in Eq. 2 under the current r_{θ}, and then approximating μ using dynamic programming. Next, we compute the state visitation distribution of the expert using a procedure which we denote as State-Visitations-Traj. This can be done either empirically, by averaging the state visitation of the experts demonstrations, or by using State-Visitations-Policy if the true reward is available at meta-training time. This allows us to recover the IRL gradient according to Eq. 5, which we can then apply to compute $\phi_{\mathcal{T}}$ according to Eq. 6.

Second, we need to differentiate through this update to compute the gradient of the meta-loss in Eq. 7. Note that the meta-loss itself is the IRL loss evaluated with a different set of test demonstrations. We follow the same procedure as above to evaluate the gradient of $\mathcal{L}^{\mathcal{T},\text{test}}_{\text{IRL}}$ with respect to the post-update parameters $\phi_{\mathcal{T}}$, and then apply the chain rule to compute the meta-gradient:

$$\begin{aligned}\nabla_\theta \mathcal{L}^{\text{test}}_{\mathcal{T}}(\theta) &= \frac{\partial}{\partial \theta}(\theta - \alpha \nabla_\theta \mathcal{L}^{\text{tr}}_{\mathcal{T}}(\theta)) \frac{\partial r_{\boldsymbol{\phi}_\mathcal{T}}}{\partial \boldsymbol{\phi}_\mathcal{T}} \frac{\partial \mathcal{L}^{\text{test}}_{\mathcal{T}}}{\partial r_{\boldsymbol{\phi}_\mathcal{T}}} \\ &= \left(\mathbf{I} - \alpha \frac{\partial^2 \mathcal{L}^{\text{tr}}_{\mathcal{T}}(\theta)}{\partial \theta^2} - \alpha \frac{\partial\, r_\theta}{\partial\, \theta} D \frac{\partial\, r_\theta}{\partial\, \theta}^\top \right) \frac{\partial r_{\boldsymbol{\phi}_\mathcal{T}}}{\partial \boldsymbol{\phi}_\mathcal{T}} \frac{\partial \mathcal{L}^{\text{test}}_{\mathcal{T}}}{\partial r_{\boldsymbol{\phi}_\mathcal{T}}} \end{aligned} \tag{8}$$

where on the second line we differentiate through the MaxEnt-IRL update, and we define the $|\mathcal{S}||\mathcal{A}|$-dimensional diagonal matrix D as

$$D := \text{diag}\left(\left\{ \frac{\partial}{\partial\, r_{\theta,i}} (\mathbb{E}_\tau[\mu_\tau])_i \right\}_{i=1}^{|\mathcal{S}||\mathcal{A}|} \right).$$

A detailed derivation of this expression is provided in the supplementary Appendix A.5.
Meta-testing. Once we have acquired the meta-trained parameters θ that encode a prior over $p(\mathcal{T})$, we can leverage this prior to enable fast, few-shot IRL of novel tasks in $\{\mathcal{T}_j\ ;\ j = 1..M\}$. For each task, we first compute the state visitations from the available set of demonstrations for that task. Next, we use these state visitations to compute the gradient, which is the same as the inner loss gradient computation of the meta-training loop in Alg. 1. We apply this gradient to adapt the parameters θ to the new task. Even if the model was trained with only one inner gradient steps, we found in practice that it was beneficial to take substantially more gradient steps during meta-testing; performance continued to improve with up to 20 steps.

2.3.2 *Connection to Learning a Prior over Intent*

The objective in Eq. 6 optimizes for parameters that enable the reward function to generalize efficiently on a wide range of tasks. Intuitively, constraining the space of reward functions to lie within a few steps of gradient descent can be interpreted as expressing a "locality" prior over reward function parameters. This intuition can be made more concrete by the following analysis.

By viewing IRL as maximum likelihood estimation in a particular graphical model (Fig. 2), we can take the perspective of Grant et al. (2018a) who showed that for a linear model, fast adaptation via a few steps of gradient descent in MAML is performing MAP inference over $\boldsymbol{\phi}$, under a Gaussian prior with the mean θ and a covariance that depends on the step size, number of steps and hessian of the loss. This is based on the connection between early stopping and regularization previously discussed in Santos

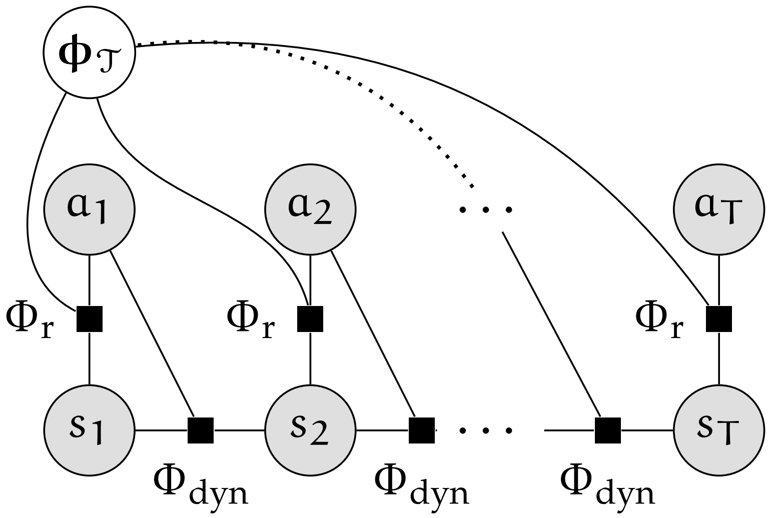


Figure 2: Our approach can be understood as approximately learning a distribution over the demonstrations τ, in the factor graph $p(\tau) = \frac{1}{Z}\prod_{t=1}^{T} \Phi_r(\boldsymbol{\phi}_{\mathcal{T}}, \mathbf{s}_t, \mathbf{a}_t)\Phi_{\text{dyn}}(\mathbf{s}_{t+1}, \mathbf{s}_t, \mathbf{a}_t)$ (above) where we learn a prior over $\phi_{\mathcal{T}}$, which during meta-test is used for MAP inference over new expert demonstrations.

(1996a), which we refer the readers to for a more detailed discussion. The interpretation of MAML as imposing a Gaussian prior on the parameters is exact in the case of a likelihood that is quadratic in the parameters (such as the log-likelihood of a Gaussian in terms of its mean). For any non-quadratic likelihood, this is an approximation in a local neighborhood around θ (i.e. up to convex quadratic approximation). In the case of complex parameterizations, such as deep function approximators, this is a coarse approximation and unlikely to be the mode of a posterior. However, we can still frame the effect of early stopping and initialization as serving as a prior in a similar way as prior work (Sjöberg and Ljung, 1995; Duvenaud et al., 2016; Grant et al., 2018a). More importantly, this interpretation hints at future extensions to our approach that could benefit from employing more fully Bayesian approaches to reward and goal inference.

2.4 EXPERIMENTS

Our evaluation seeks to answer two questions. First, we aim to test our core hypothesis that using prior task experience enables reward learning for new tasks with just a few demonstrations. Second, we compare our method with alternative approaches that make use of multi-task experience.

We test our core hypothesis by comparing learning performance on a new tasks starting from the initialization produced by MandRIL with learning a separate model for every task starting either from a random initialization or from an initialization obtained by supervised pre-training. We refer to these approaches as learning "FROM SCRATCH" and "AVERAGE GRADIENT" pretraining respectively. Our supervised pre-training baseline follows the average gradient during meta-training tasks and finetunes at meta-test time (as discussed in Section 2.3). Unlike our method, supervised pre-training does not optimize for a model that performs well under fine tuning, but does use the same prior data to pre-train. We additionally compare to pre-training on a single task as well as all the meta-training tasks.

To our knowledge, there is no prior work that addresses the specific meta-inverse reinforcement learning problem introduced in this paper. Thus, to provide a point of comparison and calibrate the difficulty of the tasks, we adapt two alternative black-box meta-learning methods to the IRL setting. The comparisons to both of the black-box methods described below evaluate the importance of incorporating the IRL gradient into the meta-learning process, rather than learning the adaptation process entirely from scratch.

DEMO CONDITIONAL MODEL: Our method implicitly conditions on the demonstrations through the gradient update. In principle, a conditional deep model with sufficient capacity could implicitly implement a similar learning rule. Thus, we consider a conditional model (often referred to as a "contextual model" (Finn et al., 2017d)), which receives the demonstration as an additional input.

RECURRENT META-LEARNER: We additionally compare to an RNN-based meta-learner (Santoro et al., 2016; Duan et al., 2017). Specifically, we implement a conditional model by feeding both images and sequences of states visited by the demonstrations to an LSTM.

We consider two environments: **(1)** an image-based navigation task with an aerial viewpoint, **(2)** a first-person navigation task in a simulated home environment with object interaction. We describe here the environments and evaluation protocol and provide detailed experimental settings and hyperparameters for both domains in Appendices A.1 and A.2.

(1) SPRITEWORLD NAVIGATION DOMAIN. Since most prior IRL work (and multi-task IRL work) studied settings where linear reward function approximators suffice (i.e., low-dimensional state spaces and hand-designed features), we design an experiment that is significantly more challenging—that requires learning rewards on raw pixels. We consider a navigation problem where we must learn a convolutional neural network that directly maps image pixels to rewards. We introduce a family of tasks called "SpriteWorld." Some example tasks are shown in Fig. 3. Tasks involve navigating to goal objects while exhibiting preference over terrain types (e.g., the agent prefers to traverse dirt tiles over

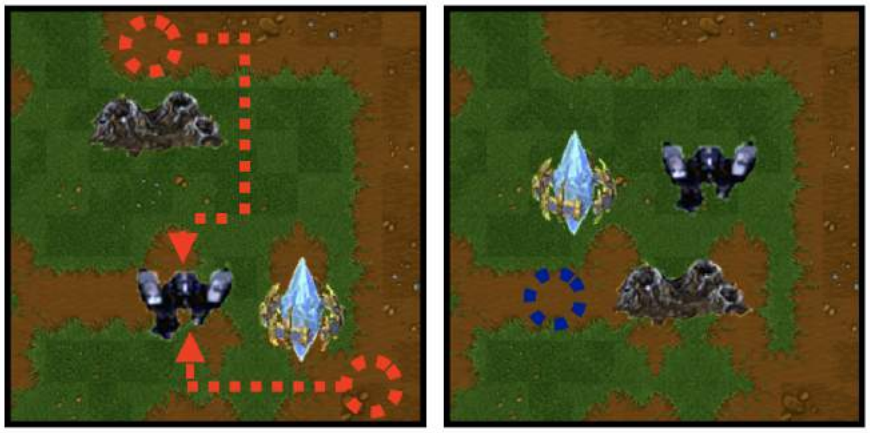


Figure 3: An example task on the SpriteWorld domain. When learning a task, the agent has access to the image (left) and demonstrations (red arrows). To evaluate learning (right), the agent is tested for its ability to recover the reward for the task when the objects have been rearranged. The reward structure we wish to capture can be illustrated by considering the initial state in blue. An policy acting optimally under a correctly inferred reward should interpret the other objects as obstacles, and prefer a path on dirt.

traversing grass tiles). At meta-test time, we provide one or a few demonstrations in a single training environment and evaluate the reward learned using these demonstrations in a new, test environment that contains the same objects as the training environment, but arranged differently. Evaluating in a new test environment is critical to measure that, after adapting to the training environment from a few demonstrations, the reward learned the correct visual cues, rather than simply memorizing the demonstration trajectory.

We generate unique tasks in this domain as follows. First, we randomly choose a set of three sprites from one hundred sprites from the original game (creating a total of 161,700 unique tasks). We randomly place these three sprites within a randomly generated terrain tiling; we designate one of the sprites to be the goal landmark of the navigation task. The other two objects are treated as obstacles for which the agent incurs a large negative reward for not avoiding. In each task, we optimize our model on a training world and generalization in a test world, as described below.

(2) SUNCG NAVIGATION DOMAIN: In addition to the SpriteWorld domain, we evaluate our approach on a first person image-based navigation task in an indoor house environment where the agent must interact with objects. We use an environment built on top of the SUNCG dataset (Song et al., 2017) which has previously been used in the context of IRL Fu et al., 2019 with language instructions. We follow a similar task setup as Fu et al. (2019), although we omit the language instructions. In this domain, we

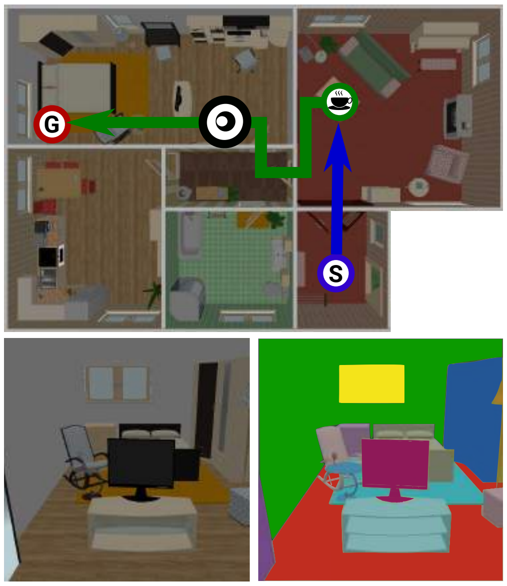


Figure 4: An example task in the SUNCG environment (top). The agent must complete either a "NAV" task (blue line) where the goal is to navigate from the start to the cup or a "PICK" task (blue + green line) where the agent must also bring the cup to the bed. The agent's observation is a panoramic first-person viewpoint (see bottom left for RGB). Following the convention in prior work Fu et al., 2019, we provide to the reward function the corresponding semantic images (bottom right). These images are 32×24 containing 61 channels corresponding to each object class.

consider tasks that can be categorized into two types.

NAVIGATION (NAV): In this task, the agent must navigate to a location in the house that corresponds either to a target object or location. For example, in the blue line of Fig. 4, the agent must navigate to the "cup" object.

PICK-AND-PLACE (PICK): In this more difficult task, the agent moves an object between two locations. For example, in Fig 4, the agent must navigate to the "cup", perform a pick action and then navigate to the "bedroom".

EVALUATION PROTOCOL: We evaluate on held-out tasks that were unseen during meta-training. In the SpriteWorld domain, we consider two settings: (1) tasks involving new combinations and placements of sprites, with sprites that were present during meta-training, and (2) tasks with combinations of unseen sprites which we refer to as "out of domain objects." For each task, we generate one environment (a set of sprite positions) along with demonstrations for adapting the reward, and generate a second environment (with new sprite positions) for evaluating the adapted reward.

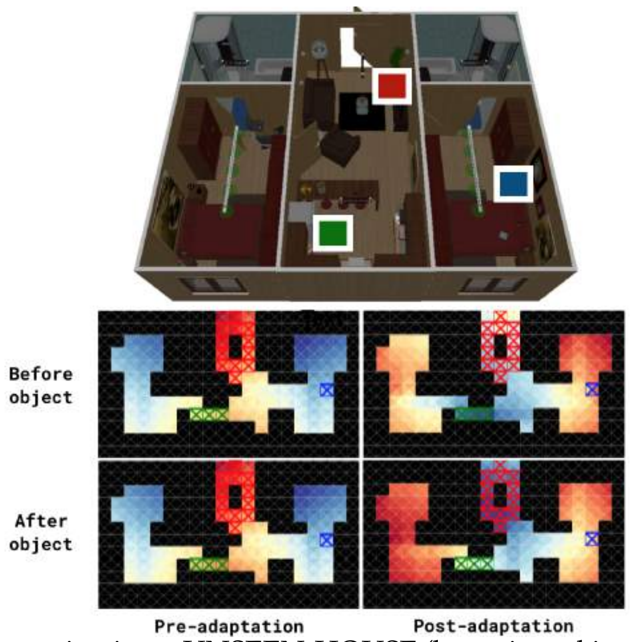


Figure 5: An example adaptation in an UNSEEN-HOUSE (best viewed in color). The agent starts in one room (blue square) and is required to pick up the vase (green square) and take it to the living room (red square). The value function (blue is high, red is low) under the learned reward (bottom) exhibits no "PICK" task structure pre-adaptation (bottom left-column). Post-adaptation (bottom right-column), the reward function successfully leads the agent to the vase (bottom figure, top-right plot) and after the pick action (bottom figure, bottom-right plot) is performed, navigates the agent to the goal location.

In the SUNCG domain, we similarly evaluate on both novel combinations of objects and locations. We follow the evaluation protocol of Fu et al. (2019) and evaluate on "TEST" tasks which consist of tasks within the same houses as training, but with novel combinations of objects and locations. In addition, we evaluate on environments which consists of new houses not in the training set. We refer to these as "UNSEEN-HOUSES." This evaluation adds complexity by testing the models ability to successfully infer rewards in an entirely new scene. In total, the dataset consists of 1413 tasks (716 PICK, 697 NAV). The meta-train set is composed of 1004 tasks, the "TEST" set contains 236 tasks, and the "UNSEEN-HOUSES" set contains 173 tasks.

EVALUATION METRICS. We measure performance using the expected value difference, which measures the sub-optimality of a policy learned under the learned reward; this is a performance metric used in prior IRL work (Levine et al., 2011; Wulfmeier et al., 2015a). The metric is computed by taking the difference between the value of the optimal policy under the learned reward and the value of the optimal policy under the true

Table 1: Success rate (%) on heldout tasks with 5 demonstrations. ManDRIL achieves consistently better performance on all task/environment types. Results are averaged over 3 random seeds.

Method	Test			Unseen Houses		
	Pick	NAV	Total	Pick	Nav	Total
Behavioral Cloning	0.4	8.2	4.3	3.7	12.0	9.4
MaxEnt IRL (avg gradient)	37.3	83.7	60.8	38.3	89.7	73.3
MaxEnt IRL (from scratch)	42.4	87.9	65.4	48.1	89.9	76.5
MandRIL(ours)	**52.3**	**90.7**	**77.3**	**56.3**	**91.0**	**82.6**
MandRIL (Pre-adaptation)	6.0	35.3	20.7	4.3	34.6	25.3

reward. On the SUNCG domain, we follow Fu et al. (2019) and report the success rate of the optimal policy under the learned reward function.

RESULTS. The results for SpriteWorld are shown in Fig. 6, which illustrate test performance with in-distribution and out-of-distribution sprites. Our approach, MandRIL, achieves consistently better performance in both settings. Most significantly, our approach performs well even with single-digit numbers of demonstrations. By comparison, alternative meta-learning methods generally overfit considerably, attaining good training performance (see Appendix. A.3 for curves) but poor test performance. Learning the reward function from scratch is in fact the most competitive baseline – as the number of demonstrations increases, simply training the reward function from scratch on the new task is the only method that matches the performance of MandRIL when provided 20 or more demonstrations. With only a few demonstrations however, MandRIL has substantially lower value difference. It is worth noting the performance of MandRIL on the out of distribution test setting (Fig. 6, bottom): although the evaluation is on new sprites, MandRIL is still able to adapt via gradient descent and exceed the performance all other methods.

In both domains, we perform a comparison to representations finetuned from a supervised pre-training phase in Fig. 6 and Table 9. We compare against an approach that follows the mean gradient across the tasks at meta-training time and is fine-tuned at meta-test time which we find consistently leads to negative transfer. We conclude that fine tuning reward functions learned in this manner is not an effective way of using prior task information. In contrast, we find that our approach, which explicitly optimizes for initial weights for fine-tuning, robustly improves performance on all task types and test settings. By visualizing the value under the learned reward function (see Fig. 5), we see that even with a small number of gradient steps, the reward function can be effectively adapted to an unseen home layout.

Note that the SUNCG task is substantially more challenging, requiring the reward function to interpret first-person images. Indeed, the pretrained MaxEntIRL algorithm that does not use meta-learning exhibits *negative* transfer, as illustrated by the lower performance of this method on all tasks as compared to the learning "FROM SCRATCH" version, which learns each task entirely from random initialization. Training from scratch is a strong baseline here, because the method still sees every single first-person image in the house – 2257.7 images on average. This provides sufficient variety to learn effective visual features in many cases. Nonetheless, our method (last row in Table 1) produces a substantial improvement, especially on the much harder "PICK" task, demonstrating that meta-learning can produce *positive* transfer even when pre-training does not.

2.5 CONCLUSION

In this work, we present an approach that enables few-shot learning of reward functions. We achieve this through a novel formulation of IRL that learns to encode common structure across tasks. Using our meta-IRL approach, we show that we can leverage data from previous tasks to effectively learn reward functions from raw pixel observations for new tasks, from only a handful of demonstrations. Our work paves the way for future work that considers unknown dynamics, or work that employs more fully probabilistic approaches to reward and goal inference.

Algorithm 1 Meta Reward and Intention Learning (MandRIL)

```
1:  Input: Set of meta-training tasks $\{\mathcal{T}\}^{\text{meta-train}}$
2:  Input: hyperparameters $\alpha, \beta$
3:  function MaxEntIRL-Grad($r_\theta$, $\mathcal{T}$, $\mathcal{D}$)                ▷ Single task update
4:      # Compute state visitations of demos
5:      $\mu_\mathcal{D}$ = State-Visitations-Traj($\mathcal{T}$, $\mathcal{D}$)
6:      # Compute Max-Ent state visitations
7:      $\mathbb{E}_\tau[\mu_\tau]$ = State-Visitations-Policy($r_\theta$, $\mathcal{T}$)
8:      # MaxEntIRL gradient (Ziebart et al., 2008b)
9:      $\frac{\partial \mathcal{L}}{\partial r_\theta} = \mathbb{E}_\tau[\mu_\tau] - \mu_\mathcal{D}$
10:     Return $\frac{\partial \mathcal{L}}{\partial r_\theta}$
11:
12: Randomly initialize $\theta$
13: while not done do
14:     Sample batch of tasks $\mathcal{T}_i \sim \{\mathcal{T}\}^{\text{meta-train}}$
15:     for all $\mathcal{T}_i$ do
16:         Sample demos $\mathcal{D}^{\text{tr}} = \{\tau_1, \ldots, \tau_K\} \sim \mathcal{T}_i$
17:         # Inner loss computation
18:         $\frac{\partial \mathcal{L}^{\text{tr}}_{\mathcal{T}_i}(\theta)}{\partial r_\theta}$ = MaxEntIRL-Grad($r_\theta$, $\mathcal{T}_i$, $\mathcal{D}^{\text{tr}}$)
19:         Compute $\nabla_\theta \mathcal{L}^{\text{tr}}_{\mathcal{T}_i}(\theta)$ from $\frac{\partial \mathcal{L}^{\text{tr}}_{\mathcal{T}_i}(\theta)}{\partial r_\theta}$
20:         Compute $\phi_{\mathcal{T}_i} = \theta - \alpha \nabla_\theta \mathcal{L}^{\text{tr}}_{\mathcal{T}_i}(\theta)$
21:         Sample demos $\mathcal{D}^{\text{test}} = \{\tau'_1, \ldots, \tau'_{K'}\} \sim \mathcal{T}_i$
22:         # Outer loss computation
23:         $\frac{\partial \mathcal{L}^{\text{test}}_{\mathcal{T}_i}}{\partial r_\theta}$ = MaxEntIRL-Grad($r_{\phi_{\mathcal{T}_i}}$, $\mathcal{T}_i$, $\mathcal{D}^{\text{test}}$))
24:         # Compute meta-gradient
25:         Compute $\nabla_\theta \mathcal{L}^{\text{test}}_{\mathcal{T}_i}$ from $\frac{\partial \mathcal{L}^{\text{test}}_{\mathcal{T}_i}}{\partial r_\theta}$ via chain rule
26:     Compute update to $\theta \leftarrow \theta - \beta \sum_i \nabla_\theta \mathcal{L}^{\text{test}}_{\mathcal{T}_i}$
```

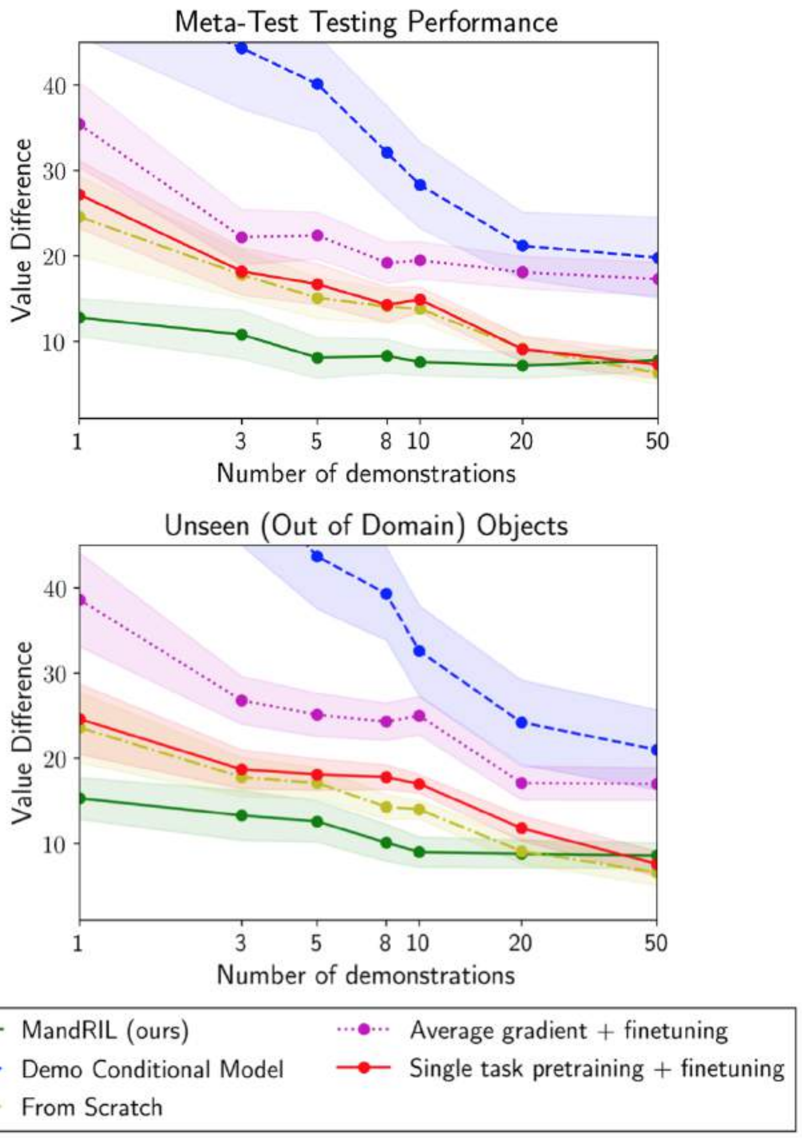


Figure 6: Meta-test performance on the SpriteWorld domain (lower is better): held-out tasks performance (top) and held-out tasks with novel sprites (bottom). The recurrent meta-learner has a value difference > 60 in both test settings. In both test settings, MandRIL achieves comparable performance to the training environment, while the other methods overfit until they receive at least 10 demonstrations (see Appendix A.3 for training environment performance). We find that pre-training on the full set of tasks leads to negative transfer, while pre-training on a single task is comparable to random initialization. ManDRIL outperforms both alternative initialization approaches, which shows that optimizing for initial weights for fine-tuning robustly improves performance. Shaded regions show 95% confidence intervals.

3

PROBABLISTIC MODEL AGNOSTIC META-LEARNING

In the last chapter, we showed how it was possible to acquire solutions to complex tasks from only a few examples by leveraging past experience to learn a "prior" over intent. The process of learning this prior entailed discovering the shared structure across different tasks from the same family, such as commonly occurring visual features or semantic cues. Algorithms that yield efficient learning, often referred to as learning-to-learn or meta-learning (Braun et al., 2010), of new tasks are a promising way to reuse related tasks. However, when the end goal of few-shot meta-learning is to learn solutions to new tasks from small amounts of data, a critical issue that must be dealt with is task *ambiguity*: even with the best possible prior, there might simply not be enough information in the examples for a new task to resolve that task with high certainty. It is therefore quite desireable to develop few-shot meta-learning methods that can propose multiple potential solutions to an ambiguous few-shot learning problem. Such a method could be used to evaluate uncertainty (by measuring agreement between the samples), perform active learning, or elicit direct human supervision about which sample is preferable. For example, in safety-critical applications, such as few-shot medical image classification, uncertainty is crucial for determining if the learned classifier should be trusted. When learning from such small amounts of data, uncertainty estimation can also help predict if additional data would be beneficial for learning and improving the estimate of the rewards. Finally, while we do not experiment with this in this paper, we expect that modeling this ambiguity will be helpful for reinforcement learning problems, where it can be used to aid in exploration.

While recognizing and accounting for ambiguity is an important aspect of the few-shot learning problem, it is challenging to model when scaling to high-dimensional data, large function approximators, and multimodal task structure. Representing distributions over functions is relatively straightforward when using simple function approximators, such as linear functions, and has been done extensively in early few-shot learning approaches using Bayesian models (Tenenbaum, 1999; Fei-Fei et al., 2003). But this problem becomes substantially more challenging when reasoning over high-dimensional function approximators such as deep neural networks, since explicitly representing expressive distributions over thousands or millions of parameters if often intractable. As a result, recent more scalable approaches to few-shot learning have focused on acquiring deterministic learning algorithms that disregard ambiguity over the underlying function. Can we develop an approach that has the benefits of both classes of few-shot learning methods – scalability and uncertainty awareness? To do so, we build upon tools in amortized variational inference for developing a probabilistic meta-learning approach.

In particular, our method builds on model-agnostic meta-learning (MAML) Finn et al., 2017a, a few shot meta-learning algorithm that uses gradient descent to adapt the model at meta-test time to a new few-shot task, and trains the model parameters at meta-training time to enable rapid adaptation, essentially optimizing for a neural network initialization that is well-suited for few shot learning. MAML can be shown to retain the generality of black-box meta-learners such as RNNs Finn and Levine, 2017, while being applicable to standard neural network architectures. Our approach extends MAML to model a distribution over prior model parameters, which leads to an appealing simple stochastic adaptation procedure that simply injects noise into gradient descent at meta-test time. The meta-training procedure then optimizes for this simple inference process to produce samples from an approximate model posterior.

The primary contribution of this paper is a reframing of MAML as a graphical model inference problem, where variational inference can provide us with a principled and natural mechanism for modeling uncertainty. Our approach enables sampling multiple potential solutions to a few-shot learning problem at meta-test time, and our experiments show that this ability can be used to sample multiple possible regressors for an ambiguous regression problem, as well as multiple possible classifiers for ambiguous few-shot attribute classification tasks. We further show how this capability to represent uncertainty can be used to inform data acquisition in a few-shot active learning problem.

3.1 RELATED WORK

Hierarchical Bayesian models are a long-standing approach for few-shot learning that naturally allow for the ability to reason about uncertainty over functions (Tenenbaum, 1999; Fei-Fei et al., 2003; Lawrence and Platt, 2004; Yu et al., 2005; Gao et al., 2008; Daumé III, 2009; Wan et al., 2012). While these approaches have been demonstrated on simple few-shot image classification datasets Lake et al., 2015, they have yet to scale to the more complex problems, such as the experiments in this paper. A number of works have approached the problem of few-shot learning from a meta-learning perspective Schmidhuber, 1987b; Hochreiter et al., 2001, including black-box Santoro et al., 2016; Duan et al., 2016; Wang et al., 2016 and optimization-based approaches Ravi and Larochelle, 2017; Finn et al., 2017a. While these approaches scale to large-scale image datasets Vinyals et al., 2016 and visual reinforcement learning problems Mishra et al., 2018, they typically lack the ability to reason about uncertainty.

Our work is most related to methods that combine deep networks and probabilistic methods for few-shot learning Edwards and Storkey, 2017; Grant et al., 2018b; Lacoste et al., 2017. One approach that considers hierarchical Bayesian models for few-shot learning is the neural statistician Edwards and Storkey, 2017, which uses an explicit task variable to model task distributions. Our method is fully model agnostic, and directly samples model weights for each task for any network architecture. Our experiments show that our approach improves on MAML Finn et al., 2017a, which outperforms the model by Edwards and Storkey (2017). Other work that considers model uncertainty in the few-shot learning setting is the LLAMA method (Grant et al., 2018b), which also builds on the MAML algorithm. LLAMA makes use of a local Laplace approximation for modeling the task parameters (post-update parameters), which introduces the need to approximate a high dimensional covariance matrix. We instead propose a method that approximately infers the pre-update parameters, which we make tractable through a choice of approximate posterior parameterized by gradient operations.

Bayesian neural networks MacKay, 1992; Hinton and Van Camp, 1993; Neal, 1995; Barber and Bishop, 1998 have been studied extensively as a way to incorporate uncertainty into deep networks. Although exact inference in Bayesian neural networks is impractical, approximations based on backpropagation and sampling Graves, 2011; Rezende et al., 2014; Hoffman et al., 2013; Blundell et al., 2015 have been effective in incorporating uncertainty into the weights of generic networks. Our approach differs from these methods in that we explicitly train a hierarchical Bayesian model over weights, where a posterior task-specific parameter distribution is inferred at meta-test time conditioned on a learned weight prior and a (few-shot) training set, while conventional Bayesian neural networks directly learn only the posterior weight distribution for a single task. Our method draws on amortized variational inference methods Kingma and Welling, 2013; Johnson et al., 2016; Shu et al., 2018 to make this possible, but the key modification is that the model and inference networks share the same parameters. The resulting method corresponds structurally to a Bayesian version of model-agnostic meta-learning Finn et al., 2017a.

3.2 PRELIMINARIES

In the meta-learning problem setting that we consider, the goal is to learn models that can learn new tasks from small amounts of data. To do so, meta-learning algorithms require a set of meta-training and meta-testing tasks drawn from some distribution $p(\mathcal{T})$. The key assumption of learning-to-learn is that the tasks in this distribution share common structure that can be exploited for faster learning of new tasks. Thus, the goal of the meta-learning process is to discover that structure. In this section, we will introduce notation and overview the model-agnostic meta-learning (MAML) algorithm Finn et al., 2017a.

Meta-learning algorithms proceed by sampling data from a given task, and splitting the sampled data into a set of a few datapoints, $\mathcal{D}^{\text{tr}}$ used for training the model and a set of datapoints for measuring whether or not training was effective, $\mathcal{D}^{\text{test}}$. This second dataset is used to measure few-shot generalization drive meta-training of the learning procedure. The MAML algorithm trains for few-shot generalization by optimizing for a set of initial parameters θ such that one or a few steps of gradient descent on $\mathcal{D}^{\text{tr}}$ achieves good performance on $\mathcal{D}^{\text{test}}$. Specifically, MAML performs the following optimization:

$$\min_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}(\theta - \alpha \nabla_{\theta} \mathcal{L}(\theta, \mathcal{D}^{\text{tr}}_{\mathcal{T}_i}), \mathcal{D}^{\text{test}}_{\mathcal{T}_i}) = \min_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}(\phi_i, \mathcal{D}^{\text{test}}_{\mathcal{T}_i})$$

where ϕ_i is used to denote the parameters updated by gradient descent and where the loss corresponds to negative log likelihood of the data. In particular, in the case of supervised classification with inputs $\{\mathbf{x}_j\}$, their corresponding labels $\{\mathbf{y}_j\}$, and a classifier f_θ, we will denote the negative log likelihood of the data under the classifier as $\mathcal{L}(\theta, \mathcal{D}) = -\sum_{(\mathbf{x}_j, \mathbf{y}_j) \in \mathcal{D}} \log p(\mathbf{y}_j | \mathbf{x}_j, \theta)$. This corresponds to the cross entropy loss function.

3.3 METHOD

Our goal is to build a meta-learning method that can handle the uncertainty and ambiguity that occurs when learning from small amounts of data, while scaling to highly-expressive function approximators such as neural networks. To do so, we set up a graphical model for the few-shot learning problem. In particular, we want a hierarchical Bayesian model that includes random variables for the prior distribution over function parameters, θ, the distribution over parameters for a particular task, ϕ_i, and the task training and test datapoints. This graphical model is illustrated in Figure 7 (left), where tasks are indexed over i and datapoints are indexed over j. We will use the shorthand $\mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}, \mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}}$ to denote the sets of datapoints $\{\mathbf{x}_{i,j}^{\text{tr}} | \, \forall\, j\}, \{\mathbf{y}_{i,j}^{\text{tr}} | \, \forall\, j\}, \{\mathbf{x}_{i,j}^{\text{test}} | \, \forall\, j\}, \{\mathbf{y}_{i,j}^{\text{test}} | \, \forall\, j\}$ and $\mathcal{D}_i^{\text{tr}}, \mathcal{D}_i^{\text{test}}$ to denote $\{\mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}\}$ and $\{\mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}}\}$.

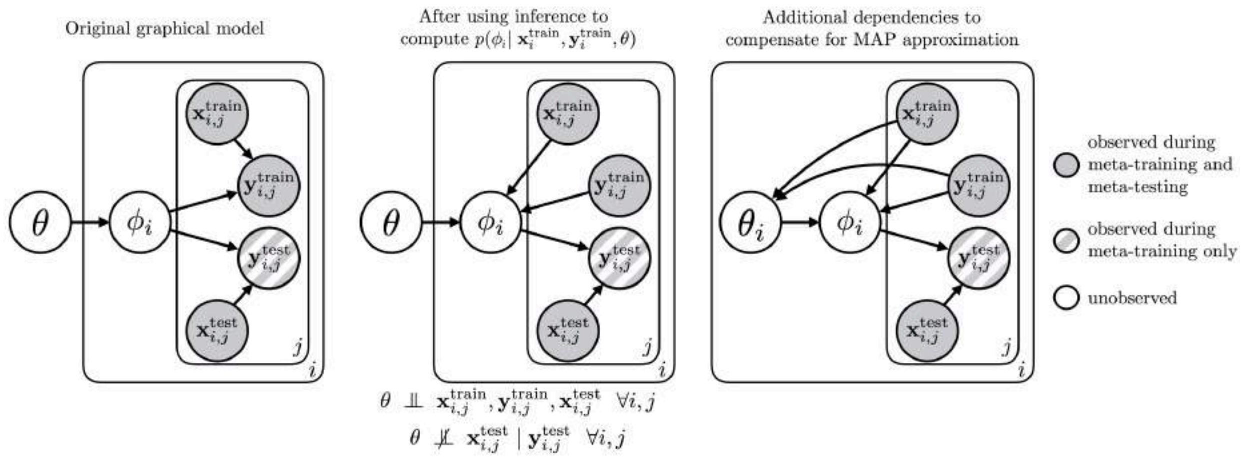

Original graphical model
After using inference to compute $p(\phi_i|\,\mathbf{x}_i^{\text{train}}, \mathbf{y}_i^{\text{train}}, \theta)$
Additional dependencies to compensate for MAP approximation
θ
ϕ_i
θ_i
$\mathbf{x}_{i,j}^{\text{train}}$
$\mathbf{y}_{i,j}^{\text{train}}$
$\mathbf{y}_{i,j}^{\text{test}}$
$\mathbf{x}_{i,j}^{\text{test}}$
j
i
observed during meta-training and meta-testing
observed during meta-training only
unobserved
$\theta \perp\!\!\!\perp \mathbf{x}_{i,j}^{\text{train}}, \mathbf{y}_{i,j}^{\text{train}}, \mathbf{x}_{i,j}^{\text{test}} \quad \forall i, j$
$\theta \not\perp\!\!\!\perp \mathbf{x}_{i,j}^{\text{test}} \mid \mathbf{y}_{i,j}^{\text{test}} \quad \forall i, j$

Figure 7: Graphical models corresponding to our approach. The original graphical model (left) is transformed into the center model after performing inference over ϕ_i. We find it beneficial to introduce additional dependencies of the prior on the training data to compensate for using the MAP estimate to approximate $p(\phi_i)$, as shown on the right.

3.3.1 *Gradient-Based Meta-Learning with Variational Inference*

In the graphical model in Figure 7, the predictions for each task are determined by the task-specific model parameters ϕ_i. At meta-test time, these parameters are influenced by the prior $p(\phi_i|\theta)$, as well as by the observed training data $\mathbf{x}^{\text{tr}}, \mathbf{y}^{\text{tr}}$. The test inputs $\mathbf{x}^{\text{test}}$ are also observed, but the test outputs $\mathbf{y}^{\text{test}}$, which need to be predicted, are not observed. Note that ϕ_i is thus independent of $\mathbf{x}^{\text{test}}$, but not of $\mathbf{x}^{\text{tr}}, \mathbf{y}^{\text{tr}}$. Therefore, posterior inference over ϕ_i must take into account both the evidence (training set) and the prior imposed by $p(\theta)$ and $p(\phi_i|\theta)$. Conventional MAML can be interpreted as approximating maximum a posteriori inference under a simplified model where $p(\theta)$ is a delta function, and inference is performed by running gradient descent on $\log p(\mathbf{y}^{\text{tr}}|\mathbf{x}^{\text{tr}}, \phi_i)$ for a fixed number of iterations starting from $\phi_i^0 = \mathbb{E}[\theta]$ Grant et al., 2018b. The corresponding distribution $p(\phi_i|\theta)$ is approximately Gaussian, with a mean that depends on the step size and number of gradient steps. When $p(\theta)$ is not deterministic, we must make a further approximation to account for the random variable θ.

One way we can do this is by using structured variational inference. In structured variational inference, we approximate the distribution over the hidden variables θ and ϕ_i for each task with some approximate distribution $q_i(\theta, \phi_i)$. There are two reasonable choices we can make for $q_i(\theta, \phi_i)$. First, we can approximate it as a product of independent marginals, according to $q_i(\theta, \phi_i) = q_i(\theta)q_i(\phi_i)$. However, this approximation does not permit uncertainty to propagate effectively from θ to ϕ_i. A more expressive approximation is the structured variational approximation $q_i(\theta, \phi_i) = q_i(\theta) q_i(\phi_i|\theta)$. We can further avoid storing a separate variational distribution $q_i(\phi_i|\theta)$ and $q_i(\theta)$ for each task $\mathcal{T}_i$ by employing an amortized variational inference technique Kingma and Welling, 2013; Johnson et al., 2016; Shu et al., 2018, where we instead set $q_i(\phi_i|\theta) = q_\psi(\phi_i|\theta, \mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}, \mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}})$, where q_ψ is defined by some function approximator with parameters ψ that takes $\mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}$ as input, and the same q_ψ is used for all tasks. Similarly, we can define $q_i(\theta)$ as $q_\psi(\theta|\mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}, \mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}})$. We can now write down the variational lower bound on the log-likelihood as

$$\log p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{test}}, \mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}) \geqslant \mathop{\mathbb{E}}_{\theta,\phi_i \sim q_\psi}\left[\log p(\mathbf{y}_i^{\text{tr}}|\mathbf{x}_i^{\text{tr}}, \phi_i) + \log p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{test}}, \phi_i) + \log p(\phi_i|\theta) + \log p(\theta)\right] +$$
$$\mathcal{H}(q_\psi(\phi_i|\theta, \mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}, \mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}})) + \mathcal{H}(q_\psi(\theta|\mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}, \mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}})).$$

The likelihood terms on the first line can be evaluated efficiently: given a sample $\theta, \phi_i \sim q(\theta, \phi_i|\mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}, \mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}})$, the training and test likelihoods simply correspond to the loss of the network with parameters ϕ_i. The prior $p(\theta)$ can be chosen to be Gaussian, with a learned mean and (diagonal) covariance to provide for flexibility to choose the prior parameters. This corresponds to a Bayesian version of the MAML algorithm. We will define these parameters as $\boldsymbol{\mu}_\theta$ and $\boldsymbol{\sigma}_\theta^2$. Lastly, $p(\phi_i|\theta)$ must be chosen. This choice is more delicate. One way to ensure a tractable likelihood is to use a Gaussian with mean θ. This choice is reasonable, because it encourages ϕ_i to stay close to the prior parameters ϕ_i, but we will see in the next section how a more expressive implicit conditional can be obtained using gradient descent, resulting in a procedure that more closely resembles the original MAML algorithm while still modeling the uncertainty. Lastly, we must choose a form for the inference networks $q_\psi(\phi_i|\theta, \mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}, \mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}})$ and $q_\psi(\theta|\mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}, \mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}})$. They must be chosen so that their entropies on the second line of the above equation are tractable. Furthermore, note that both of these distributions model very high-dimensional random variables: a deep neural network can have hundreds of thousands or millions of parameters. So while we can use an arbitrary function approximator, we would like to find a scalable solution.

One convenient solution is to allow q_ψ to reuse the learned mean of the prior $\boldsymbol{\mu}_\theta$. We observe that adapting the parameters with gradient descent is a good way to update them to a given training set $\mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}$ and test set $\mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}}$, a design decision similar to one made by Fortunato et al. (2017). We propose an inference network of the form

$$q_\psi(\theta|\mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}, \mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}}) = \mathcal{N}(\boldsymbol{\mu}_\theta + \boldsymbol{\gamma}_q \nabla_{\boldsymbol{\mu}_\theta} \log p(\mathbf{y}_i^{\text{tr}}|\mathbf{x}_i^{\text{tr}}, \boldsymbol{\mu}_\theta) + \boldsymbol{\gamma}_q \nabla_{\boldsymbol{\mu}_\theta} \log p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{test}}, \boldsymbol{\mu}_\theta); \mathbf{v}_q),$$

where $\mathbf{v}_q$ is a learned (diagonal) covariance, and the mean has an additional parameter beyond $\boldsymbol{\mu}_\theta$, which is a "learning rate" vector $\boldsymbol{\gamma}_q$ that is pointwise multiplied with the gradient. While this choice may at first seem arbitrary, there is a simple intuition: the inference network should produce a sample of θ that is close to the posterior $p(\theta|\mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}, \mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}})$. A reasonable way to arrive at a value of θ close to this posterior is to adapt it to *both* the training set and test set.[1] Note that this is only done during meta-training. It remains to choose $q_\psi(\phi_i|\theta, \mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}, \mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}})$, which can also be formulated as a conditional Gaussian with mean given by applying gradient descent.

Although this variational distribution is substantially more compact in terms of parameters than a separate neural network, it only provides estimates of the posterior during meta-training. At meta-test time, we must obtain the posterior $p(\phi_i|\mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}, \mathbf{x}_i^{\text{test}})$, without access to $\mathbf{y}_i^{\text{test}}$. We can train a separate set of inference networks to perform this operation, potentially also using gradient descent within the inference network. However, these networks do not receive any gradient information during meta-training, and may not work well in practice. In the next section we propose an even simpler and more practical approach that uses only a single inference network during meta-training, and none during meta-testing.

3.3.2 *Probabilistic Model-Agnostic Meta-Learning Approach with Hybrid Inference*

To formulate a simpler variational meta-learning procedure, we recall the probabilistic interpretation of MAML: as discussed by Grant et al. (2018b), MAML can be interpreted as approximate inference for the posterior $p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}, \mathbf{x}_i^{\text{test}})$ according to

$$p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}, \mathbf{x}_i^{\text{test}}) = \int p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{test}}, \phi_i) p(\phi_i|\mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}, \theta) d\phi_i \approx p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{test}}, \phi_i^\star), \tag{9}$$

where we use the maximum a posteriori (MAP) value $\phi_i^\star$. It can be shown that, for likelihoods that are Gaussian in ϕ_i, gradient descent for a fixed number of iterations using $\mathbf{x}_i^{\text{tr}}$, $\mathbf{y}_i^{\text{tr}}$ corresponds exactly to maximum a posteriori inference under a Gaussian prior $p(\phi_i|\theta)$ Santos, 1996b. In the case of non-Gaussian likelihoods, the equivalence is only locally approximate, and the exact form of the prior $p(\phi_i|\theta)$ is intractable. However, in practice this implicit prior can actually be preferable to an explicit (and simple) Gaussian prior, since it incorporates the rich nonlinear structure of the neural network parameter manifold, and produces good performance in practice Finn et al., 2017a; Grant et al., 2018b. We can interpret this MAP approximation as inferring an approximate posterior

1 In practice, we can use multiple gradient steps for the mean, but we omit this for notational simplicity.

Algorithm 2 Meta-training, differences from MAML in red

```
Require: $p(\mathcal{T})$: distribution over tasks
 1: initialize $\Theta := \{\boldsymbol{\mu}_\theta, \boldsymbol{\sigma}^2_\theta, \mathbf{v}_q, \boldsymbol{\gamma}_p, \boldsymbol{\gamma}_q\}$
 2: while not done do
 3:     Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
 4:     for all $\mathcal{T}_i$ do
 5:         $\mathcal{D}^{tr}, \mathcal{D}^{test} = \mathcal{T}_i$
 6:         Evaluate $\nabla_{\boldsymbol{\mu}_\theta} \mathcal{L}(\boldsymbol{\mu}_\theta, \mathcal{D}^{test})$
 7:         Sample $\theta \sim q = \mathcal{N}(\boldsymbol{\mu}_\theta - \boldsymbol{\gamma}_q \nabla_{\boldsymbol{\mu}_\theta} \mathcal{L}(\boldsymbol{\mu}_\theta, \mathcal{D}^{test}), \mathbf{v}_q)$
 8:         Evaluate $\nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{tr})$
 9:         Compute adapted parameters with gradient descent:
            $\phi_i = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{tr})$
10:     Let $p(\theta|\mathcal{D}^{tr}) = \mathcal{N}(\boldsymbol{\mu}_\theta - \boldsymbol{\gamma}_p \nabla_{\boldsymbol{\mu}_\theta} \mathcal{L}(\boldsymbol{\mu}_\theta, \mathcal{D}^{tr}), \boldsymbol{\sigma}^2_\theta))$
11:     Compute $\nabla_\Theta \big( \sum_{\mathcal{T}_i} \mathcal{L}(\phi_i, \mathcal{D}^{test})$
        $+ D_{KL}(q(\theta|\mathcal{D}^{test}) \,||\, p(\theta|\mathcal{D}^{tr})) \big)$
12:     Update $\Theta$ using Adam
```

Algorithm 3 Meta-testing

```
Require: training data $\mathcal{D}^{tr}_{\mathcal{T}}$ for new task $\mathcal{T}$
Require: learned $\Theta$
 1: Sample $\theta$ from the prior $p(\theta|\mathcal{D}^{tr})$
 2: Evaluate $\nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{tr})$
 3: Compute adapted parameters with gradient descent:
    $\phi_i = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{tr})$
```

on ϕ_i of the form $p(\phi_i|\mathbf{x}_i^{tr}, \mathbf{y}_i^{tr}, \theta) \approx \delta(\phi_i = \phi_i^\star)$, where $\phi_i^\star$ is obtained via gradient descent on the training set $\mathbf{x}_i^{tr}, \mathbf{y}_i^{tr}$ starting from θ. Incorporating this approximate inference procedure transforms the graphical model in Figure 7 (a) into the one in Figure 7 (b), where there is now a factor over $p(\phi_i|\mathbf{x}_i^{tr}, \mathbf{y}_i^{tr}, \theta)$. While this is a crude approximation to the likelihood, it provides us with an empirically effective and simple tool that greatly simplifies the variational inference procedure described in the previous section, in the case where we aim to model a distribution over the global parameters $p(\theta)$. After using gradient descent to estimate $p(\phi_i \mid \mathbf{x}_i^{tr}, \mathbf{y}_i^{tr}, \theta)$, the graphical model is transformed into the model shown in the center of Figure 7. Note that, in this new graphical model, the global parameters θ are independent of $\mathbf{x}^{tr}$ and $\mathbf{y}^{tr}$ and are independent of $\mathbf{x}^{test}$ when $\mathbf{y}^{test}$ is not observed. Thus, we can now write down a variational lower bound for the logarithm of the *approximate* likelihood, which is given by

$$\log p(\mathbf{y}_i^{test}|\mathbf{x}_i^{test}, \mathbf{x}_i^{tr}, \mathbf{y}_i^{tr}) \geqslant E_{\theta \sim q_\psi} \left[\log p(\mathbf{y}_i^{test}|\mathbf{x}_i^{test}, \phi_i^\star) + \log p(\theta) \right] + \mathcal{H}(q_\psi(\theta|\mathbf{x}_i^{test}, \mathbf{y}_i^{test})).$$

In this bound, we essentially perform approximate inference via MAP on ϕ_i to obtain $p(\phi_i|\mathbf{x}_i^{tr}, \mathbf{y}_i^{tr}, \theta)$, and use the variational distribution for θ only. Note that $q_\psi(\theta|\mathbf{x}_i^{test}, \mathbf{y}_i^{test})$ is not conditioned on the training set $\mathbf{x}_i^{tr}, \mathbf{y}_i^{tr}$ since θ is independent of it in the transformed graphical model. Analogously to the previous section, the inference network is given by

$$q_\psi(\theta|\mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}}) = \mathcal{N}(\boldsymbol{\mu}_\theta + \boldsymbol{\gamma}_q \nabla \log p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{test}}, \boldsymbol{\mu}_\theta); \mathbf{v}_q).$$

To evaluate the variational lower bound during training, we can use the following procedure: first, we evaluate the mean by starting from $\boldsymbol{\mu}_\theta$ and taking one (or more) gradient steps on $\log p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{test}}, \theta_{\text{current}})$, where θ_{current} starts at $\boldsymbol{\mu}_\theta$. We then add noise with variance $\mathbf{v}_q$, which is made differentiable via the reparameterization trick Kingma and Welling, 2013. We then take additional gradient steps on the training likelihood $\log p(\mathbf{y}_i^{\text{tr}}|\mathbf{x}_i^{\text{tr}}, \theta_{\text{current}})$. This accounts for the MAP inference procedure on ϕ_i. Training of $\boldsymbol{\mu}_\theta$, $\boldsymbol{\sigma}_\theta^2$, and $\mathbf{v}_q$ is performed by backpropagating gradients through this entire procedure with respect to the variational lower bound, which includes a term for the likelihood $\log p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{test}}, \mathbf{x}^{\text{tr}}, \mathbf{y}^{\text{tr}}, \phi_i^\star)$ and the KL-divergence between the sample $\theta \sim q_\psi$ and the prior $p(\theta)$. This meta-training procedure is detailed in Algorithm 2.

At meta-test time, the inference procedure is much simpler. The test labels are not available, so we simply sample $\theta \sim p(\theta)$ and perform MAP inference on ϕ_i using the training set, which corresponds to gradient steps on $\log p(\mathbf{y}_i^{\text{tr}}|\mathbf{x}_i^{\text{tr}}, \theta_{\text{current}})$, where θ_{current} starts at the sampled θ. This meta-testing procedure is detailed in Algorithm 3.

3.3.3 *Adding Additional Dependencies*

In the transformed graphical model, the training data $\mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}$ and the prior θ are conditionally independent. However, since we have only a crude approximation to $p(\phi_i \mid \mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}, \theta)$, this independence often doesn't actually hold. We can allow the model to compensate for this approximation by additionally conditioning the learned prior $p(\theta)$ on the training data. In this case, the learned "prior" has the form $p(\theta_i|\mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}})$, where θ_i is now task-specific, but with global parameters $\boldsymbol{\mu}_\theta$ and $\boldsymbol{\sigma}_\theta^2$. We thus obtain the modified graphical model in Figure 7 (c). Similarly to the inference network q_ψ, we parameterize the learned prior as follows:

$$p(\theta_i|\mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}}) = \mathcal{N}(\boldsymbol{\mu}_\theta + \boldsymbol{\gamma}_p \nabla \log p(\mathbf{y}_i^{\text{tr}}|\mathbf{x}_i^{\text{tr}}, \boldsymbol{\mu}_\theta); \boldsymbol{\sigma}_\theta^2).$$

With this new form for distribution over θ, the variational training objective uses the likelihood term $\log p(\theta_i|\mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}})$ in place of $\log p(\theta)$, but otherwise is left unchanged. At test time, we sample from $\theta \sim p(\theta|\mathbf{x}_i^{\text{tr}}, \mathbf{y}_i^{\text{tr}})$ by first taking gradient steps on $\log p(\mathbf{y}_i^{\text{tr}}|\mathbf{x}_i^{\text{tr}}, \theta_{\text{current}})$, where θ_{current} is initialized at $\boldsymbol{\mu}_\theta$, and then adding noise with variance $\boldsymbol{\sigma}_\theta^2$. Then, we proceed as before, performing MAP inference on ϕ_i by taking additional gradient steps on $\log p(\mathbf{y}_i^{\text{tr}}|\mathbf{x}_i^{\text{tr}}, \theta_{\text{current}})$ initialized at the sample θ. In our experiments, we find that this more expressive distribution often leads to better performance.

3.4 EXPERIMENTS

The goal of our experimental evaluation is to answer the following questions: (1) can our approach enable sampling from the distribution over potential functions underlying the training data?, (2) does our approach improve upon the MAML algorithm when there is ambiguity over the class of functions?, and (3) can our approach scale to deep convolutional networks? We study two illustrative toy examples and a realistic ambiguous few-shot image classification problem. For the both experimental domains, we compare MAML to our probabilistic approach. We will refer to our version of MAML as a PLATIPUS (Probabilistic LATent model for Incorporating Priors and Uncertainty in few-Shot learning), due to its unusual combination of two approximate inference methods: amortized inference and MAP. Both PLATIPUS and MAML use the same neural network architecture and the same number of inner gradient steps. We additionally provide a comparison on the MiniImagenet benchmark and specify the hyperparameters in the supplementary Appendix B.

ILLUSTRATIVE 5-SHOT REGRESSION. In this 1D regression problem, different tasks correspond to different underlying functions. Half of the functions are sinusoids, and half are lines, such that the task distribution is clearly multimodal. The sinusoids have amplitude and phase uniformly sampled from the range $[0.1, 5]$ and $[0, \pi]$, and the lines have the slope and intercept sampled in the range $[-3, 3]$. The input domain is uniform on $[-5, 5]$, and Gaussian noise with a standard deviation of 0.3 is added to the labels. We trained both MAML and PLATIPUS for 5-shot regression. In Figure 8, we show the qualitative performance of both methods, where the ground truth underlying function is shown in gray and the datapoints in $\mathcal{D}^{tr}$ are shown as purple triangles. We show the function f_{ϕ_i} learned by MAML in black. For PLATIPUS, we sample 10 sets of parameters from $p(\phi_i|\theta)$ and plot the resulting functions in different colors. In the top row, we can see that PLATIPUS allows the model to effectively reason over the set of functions underlying the provided datapoints, with increased variance in parts of the function where there is more uncertainty. Further, we see that PLATIPUS is able to capture the multimodal structure, as the curves are all linear or sinusoidal.

A particularly useful application of uncertainty estimates in few-shot learning is estimating when more data would be helpful. In particular, seeing a large variance in a particular part of the input space suggests that more data would be helpful for learning the function in that part of the input space. On the bottom of Figure 8, we show the results for a single task at meta-test time with increasing numbers of training datapoints.

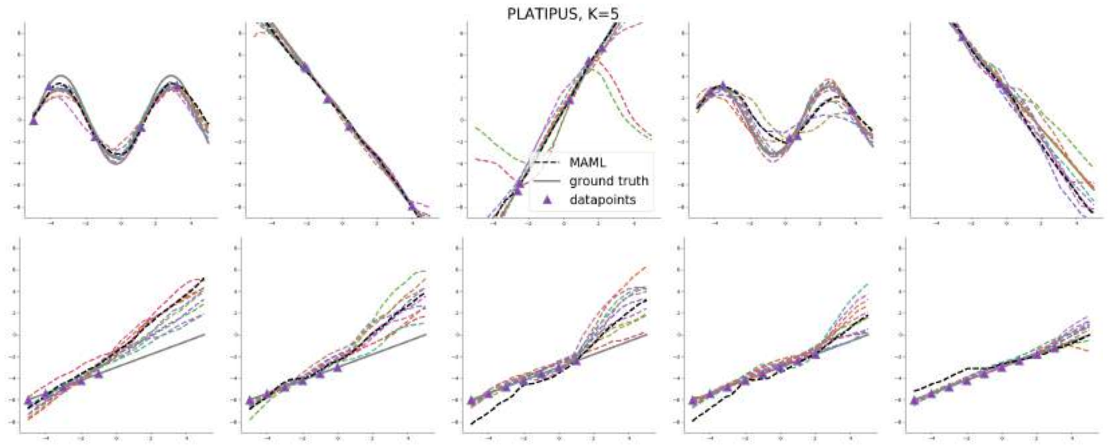


Figure 8: Samples from PLATIPUS trained for 5-shot regression, shown as colored dotted lines. The tasks consist of regressing to sinusoid and linear functions, shown in gray. MAML, shown in black, is a deterministic procedure and hence learns a single function, rather than reasoning about the distribution over potential functions. As seen on the bottom row, even though PLATIPUS is trained for 5-shot regression, it can effectively reason over its uncertainty when provided variable numbers of datapoints at test time (left vs. right).

Even though the model was only trained on training set sizes of 5 datapoints, we observe that PLATIPUS is able to effectively reduce its uncertainty as more and more datapoints are available. This suggests that the uncertainty provided by PLATIPUS can be used for approximately gauging when more data would be helpful for learning a new task.

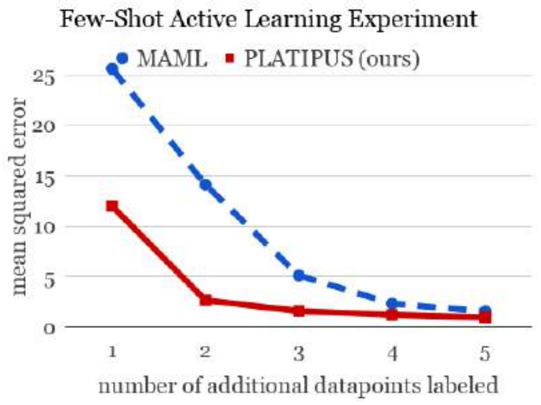


Figure 10: Active learning performance on regression after up to 5 selected datapoints. PLATIPUS can use it's uncertainty estimation to quickly decrease the error, while selecting datapoints randomly and using MAML leads to slower learning.

ACTIVE LEARNING WITH REGRESSION. To further evaluate the benefit of modeling ambiguity, we now consider an active learning experiment. In particular, the model can choose the datapoints that it wants labels for, with the goal of reaching good performance with a minimal number of additional datapoints. We performed this evaluation in the simple regression setting described previously. Models were given five initial datapoints within a constrained region of the input space. Then, each model selects up to 5 additional datapoints to be labeled. PLATIPUS chose each datapoint sequentially, choosing the point with maximal

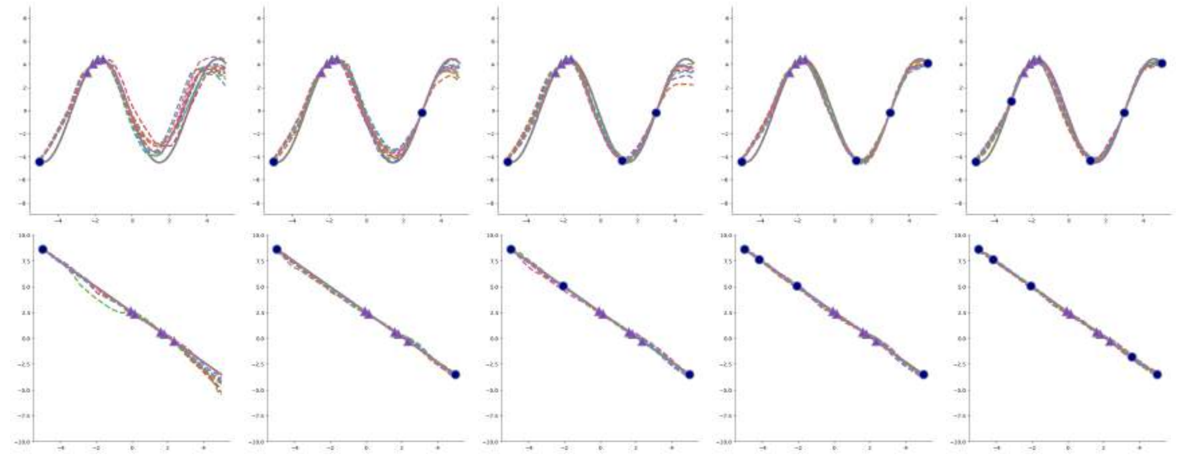


Figure 9: Qualitative examples from active learning experiment where the 5 provided datapoints are from a small region of the input space (shown as purple triangles), and the model actively asks for labels for new datapoints (shown as blue circles) by choosing datapoints with the largest variance across samples. The model is able to effectively choose points that leads to accurate predictions with only a few extra datapoints.

variance across the sampled regressors; MAML selected datapoints randomly, as it has no mechanism to model ambiguity. As seen in Figure 10, PLATIPUS is able to reduce its regression error to a much greater extent when given one to three additional queries, compared to MAML. We show qualitative results in Figure 9.

ILLUSTRATIVE 1-SHOT 2D CLASSIFICATION. Next, we study a simple binary classification task, where there is a particularly large amount of ambiguity surrounding the underlying function: learning to learn from a single positive example. Here, the tasks consist of classifying datapoints in 2D within the range $[0, 5]$ with a circular decision boundary, where points inside the decision boundary are positive and points outside are negative. Different tasks correspond to different locations and radii of the decision boundary, sampled at uniformly at random from the ranges $[1.0, 4.0]$ and $[0.1, 2.0]$ respectively. Following Grant et al. (2017), we train both MAML and PLATIPUS with $\mathcal{D}^{\text{tr}}$ consisting of a single positive example and $\mathcal{D}^{\text{test}}$ consisting of both positive and negative examples. We plot the results using the same scheme as before, except that we plot the decision boundary (rather than the regression function) and visualize the single positive datapoint with a green plus. As seen in Figure 11, we see that PLATIPUS captures a broad distribution over possible decision boundaries, all of which are roughly circular. MAML provides a single decision boundary of average size.

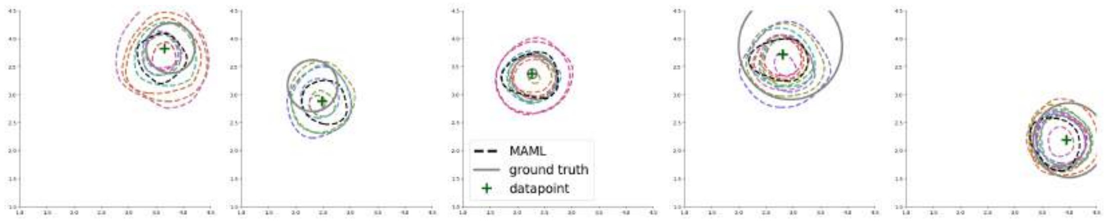


Figure 11: Samples from PLATIPUS for 1-shot classification, shown as colored dotted lines. The 2D classification tasks all involve circular decision boundaries of varying size and center, shown in gray. MAML, shown in black, is a deterministic procedure and hence learns a single function, rather than reasoning about the distribution over potential functions.

AMBIGUOUS IMAGE CLASSIFICATION. The ambiguity illustrated in the previous settings is common in real world tasks where images can share multiple attributes. We study an ambiguous extension to the celebA attribute classification task. Our meta-training dataset is formed by sampling two attributes at random to form a positive class and taking the same number of random examples without either attribute to from the negative classes. To evaluate the ability to capture multiple decision boundaries while simultaneously obtaining good performance, we evaluate our method as follows: We sample from a test set of three attributes and a corresponding set of images with those attributes. Since the tasks involve classifying images that have two attributes, this task is ambiguous, and there are three possible combinations of two attributes that explain the training set. We sample models from our prior as described in Section 6.4 and assign each of the sampled models to one of the three possible tasks based on its log-likelihood. If each of the three possible tasks is assigned a nonzero number of samples, this means that the model effectively covers all three possible modes that explain the ambiguous training set. We can measure coverage and accuracy from this protocol. The coverage score indicates the average number of tasks (between 1 and 3) that receive at least one sample for each ambiguous training set, and the accuracy score is the average number of correct classifications on these tasks (according to the sampled models assigned to them). A highly random method will achieve good coverage but poor accuracy, while a deterministic method will have a coverage of 1. We additionally compute the log-likelihood across the ambiguous tasks which compares each method's ability to model all of the "modes". As is standard in amortized variational inference (e.g., with VAEs), we put a multiplier β in front of the KL-divergence against the prior Higgins et al., 2017 in Algorithm 2. We find that larger values result in more diverse samples, at a modest cost in performance, and therefore report two different values of β to illustrate this tradeoff.

Our results are summarized in Table 2 and Fig. 12. Our method attains better log-likelihood, and a comparable accuracy compared to standard MAML. More importantly,

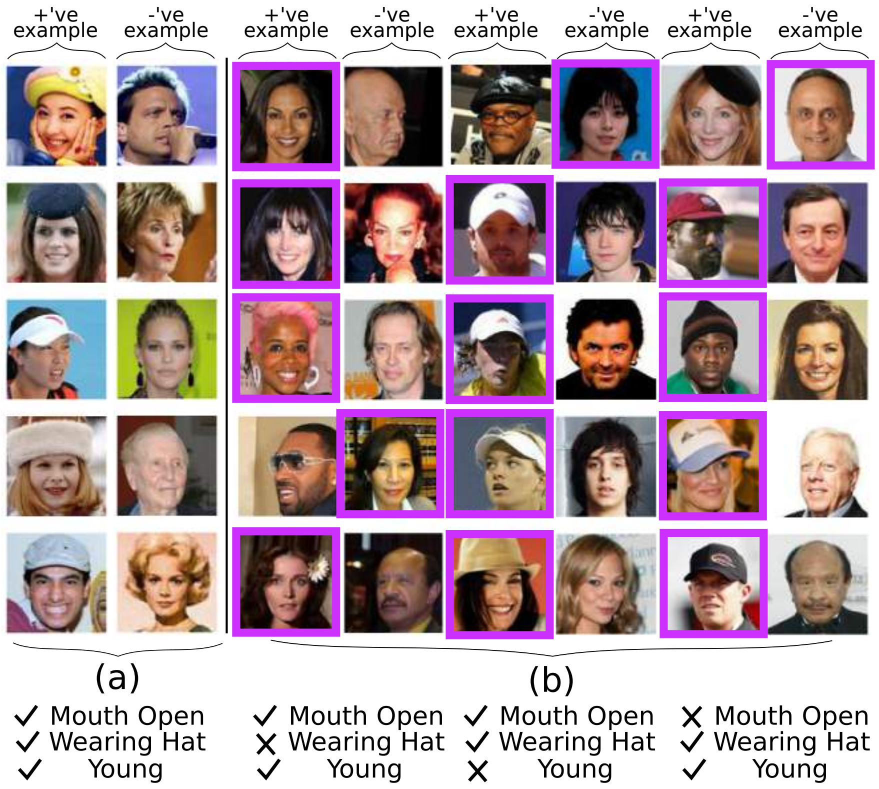


Figure 12: Sampled classifiers for apromaml/n ambiguous meta-test task. In the meta-test training set (a), PLATIPUS observes five positives that share three attributes, and five negatives. A classifier that uses *any* two attributes can correctly classify the training set. On the right (b), we show the three possible two-attribute tasks that the training set can correspond to, and illustrate the labels (positive indicated by purple border) predicted by the best sampled classifier for that task. We see that different samples can effectively capture the three possible explanations, with some samples paying attention to hats (2nd and 3rd column) and others not (1st column).

deterministic MAML only ever captures one mode for each ambiguous task, where the maximum is three. Our method on average captures closer to two modes on average. The qualitative analysis in Figure 12 illustrates an example ambiguous training set, example images for the three possible two-attribute pairs that can correspond to this training set, and the classifications made by different sampled classifiers trained on the ambiguous training set. Note that the different samples each pay attention to different attributes, indicating that PLATIPUS is effective at capturing the different modes of the task.

Ambiguous celebA (5-shot)			
	Accuracy	Coverage (max=3)	Average NLL
MAML	**89.00 ± 1.78%**	1.00 ± 0.0	0.73 ± 0.06
MAML + noise	84.3 ± 1.60 %	1.89 ± 0.04	0.68 ± 0.05
PLATIPUS (ours) (KL weight = 0.05)	**88.34 ± 1.06** %	1.59 ± 0.03	0.67± 0.05
PLATIPUS (ours) (KL weight = 0.15)	**87.8 ± 1.03** %	**1.94 ± 0.04**	**0.56 ± 0.04**

Table 2: Our method covers almost twice as many modes compared to MAML, with comparable accuracy. MAML + noise is a method that adds noise to the gradient, but does not perform variational inference. This improves coverage, but results in lower accuracy average log likelihood. We bold results above the highest confidence interval lowerbound.

3.5 DISCUSSION AND FUTURE WORK

We introduced an algorithm for few-shot meta-learning that enables simple and effective sampling of models for new tasks at meta-test time. Our algorithm, PLATIPUS, adapts to new tasks by running gradient descent with injected noise. During meta-training, the model parameters are optimized with respect to a variational lower bound on the likelihood for the meta-training tasks, so as to enable this simple adaptation procedure to produce approximate samples from the model posterior when conditioned on a few-shot training set. This approach has a number of benefits. The adaptation procedure is exceedingly simple, and the method can be applied to any standard model architecture. The algorithm introduces a modest number of additional parameters: besides the initial model weights, we must learn a variance on each parameter for the inference network and prior, and the number of parameters scales only linearly with the number of model weights. Our experimental results show that our method can be used to effectively sample diverse solutions to both regression and classification tasks at meta-test time, including with task families that have multi-modal task distributions. We additionally showed how our approach can be applied in settings where uncertainty can directly guide data acquisition, leading to better few-shot active learning.

Although our approach is simple and broadly applicable, it has potential limitations that could be addressed in future work. First, the current form of the method provides a relatively impoverished estimator of posterior variance, which might be less effective at gauging uncertainty in settings where different tasks have different degrees of ambiguity. In such settings, making the variance estimator dependent on the few-shot training set might produce better results, and investigating how to do this in a parameter efficient manner would be an interesting direction for future work. Another exciting direction for future research would be to study how our approach could be applied in RL settings for acquiring structured, uncertainty-guided exploration strategies in meta-RL problems.

4

LIFELONG LEARNING OF RESET-FREE SKILLS

In the Chapters 2 and 3, we demonstrated the potential of reusing related tasks to accelerate learning and developed algorithsm for handling uncertainty. This direction brings us closes to the goal of having adaptive, generalist agents that learn, but there still important practical challenges remain when tasked with real world learning. For reinforcement learning (RL) methods to be successfully deployed in the real world, they must solve a number of distinct challenges. First, agents that learn in real world settings, such as robotics, must contend with non-episodic learning processes: when the agent attempts the task and fails, it must start from that resulting failure state for the next attempt. Otherwise, it would require a manually-provided "reset," which is easy in simulated environments, but takes considerable human effort in the real world, thereby reducing autonomy – the very thing that makes RL appealing. Second, complex and temporally extended behaviors can be exceedingly hard to learn with naïve exploration (Kearns and Singh, 2002; Kakade et al., 2003). Agents that are equipped with a repertoire of skills or primitives (e.g., options (Sutton et al., 1999)) could master such temporally extended tasks much more efficiently.

These two problems may on the surface seem unrelated. However, the non-episodic learning problem may be addressed by learning additional skills for resetting the system. Could these same skills also provide a natural way to accelerate learning of downstream tasks? This is the question we study in this paper. Our goal is to understand how diverse "reset skills" can simultaneously enable non-episodic reset-free learning and, in the process, acquire useful primitives for accelerating downstream reinforcement learning.

Prior works have explored automated skill learning using intrinsic, unsupervised reward objectives (Gregor et al., 2016; Eysenbach et al., 2017; Sharma et al., 2019), acquiring behaviors that can then be used to solve downstream tasks with user-specified objectives efficiently, often by employing hierarchical methods (Dayan and Hinton, 1993; Sutton et al., 1999; Dietterich, 2000). These skill discovery methods assume the ability to reset to initial states during this "pre-training" phase, and even then face a challenging optimization problem: without any additional manual guidance, the space of potential behaviors is so large that practically useful skill repertoires can only be discovered in relatively simple and low dimensional domains. Therefore, we need a compromise – a more "focused" method that is likely to produce skills that are relevant for downstream problems, without requiring these skills to be defined manually.

In this paper, we study how these issues can be addressed together with removing manual resets. While the problems of skill discovery and reset-free learning may at first appear unrelated, we observe that reset-free learning of a given task can be enhanced by access to varied and diverse "reset skills," which force the task policy to succeed from a variety of starting states. At the same time, the skills can be forced to acquire more complex behaviors, through an objective that encourages them to discover states that are challenging for the task policy. This means that the same mechanism that can address the "reset" problem can also serve as a skill discovery method. Performing a single given task can be viewed as a "funneling" process: transitioning from a wide distribution over initial conditions to a narrower distribution over states where a given task has been completed successfully. Reversing this process therefore can provide coverage over a broader range of states, while still keeping this exploration process grounded to situations from which the initial task is solvable.

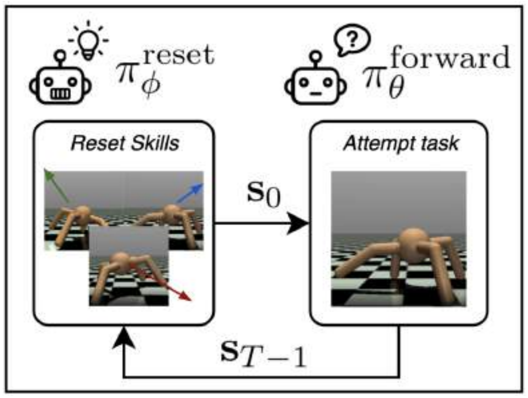


Figure 13: An outline of our approach. A reset policy $\pi_{\phi}^{\text{reset}}$ provides a state $\mathbf{s}_0$ to a forward policy π_θ to learn a task using a learned skill. Upon completion, the final states, $\mathbf{s}_{T-1}$ is reset with another selected skill. In real world settings, where all learning is continuous and sample efficiency is critical, we present a method that leverages the insight that the non-episodic learning problem can be addressed by learning a set of skills of resetting the system. We additionally show that these skills can be used to accelerate downstream learning.

The main contribution of this paper is a "reset game" that implements this idea as a general-sum game between two players: a task policy player that attempts to perform a given task, and a reset policy player, corresponding to a repertoire of distinct skills, that attempts to perturb the state to make the task harder for the task policy while *simultaneously* producing a diverse set of skills (see Figure 13). The result of this process is a setting where the challenge of learning without resets is instead viewed as an opportunity to learn a diverse set of behavioral primitives, which can then be used to accelerate downstream reinforcement learning, both removing the need for human-provided resets and acquiring skills that can be used for downstream tasks. We show that by grounding the skill learning problem in this way, we can design a method that (1) enables an agent to learn a task without requiring oracle resets, which is a non-trivial requirement in real-world applications, and (2) learns a broader set of skills compared to prior unsupervised approaches, which substantially improves effectiveness on down-stream long-horizon tasks. We demonstrate the efficacy of our approach both on previously studied robotics-themed reset-free RL problems, and tasks that reflect domains studied in prior unsupervised skill learning work, demonstrating both improved reset-free learning performance and improved performance on downstream tasks with hierarchical controllers.

4.1 RELATED WORK

Learning without access to resets has been studied in prior work with a focus on automation, safety, and learning compound controllers (Han et al., 2015; Eysenbach et al., 2017; Chatzilygeroudis et al., 2018). Eysenbach et al. (2017) propose to learn a reset controller for safe learning, but assume access to ground truth rewards and an oracle function that can detect resets. Closely related to our work, Zhu et al. (2020a) learn a perturbation controller to handle the reset-free setting, but importantly does not learn a set of skills. In contrast, we propose to learn a broad set of skills for performing resets, which are "anchored" to a specific forward RL task by requiring that the reset skills produce states that are diverse and challenging. We show that this anchoring leads to skills more suited for downstream learning, while improving reset-free performance.

Autonomous acquisition of skills is related to the learning with intrinsic motivation, which considers the unsupervised setting where an agent must learn without extrinsic reward (Schmidhuber, 1991; Chentanez et al., 2005; Klyubin et al., 2005; Barto, 2013; Baranes and Oudeyer, 2013). In the direction of curiosity-driven exploration, recent work has used state novelty to reward an agent Bellemare et al., 2016; Tang et al., 2017; Pathak et al., 2017. Similarly, other work has instead used the empowerment maximization principle Salge et al., 2014 to define mutual information based objectives from which to learn behavioral primitives Mohamed and Rezende, 2015; Gregor et al., 2016; Florensa et al., 2017; Eysenbach et al., 2018; Sharma et al., 2019. Our method incorporates unsupervised skill discovery into reset-free learning, building on mutual information methods for skill discovery. In contrast to these prior methods, our approach uses a forward task to implicitly "anchor" the skills. We show experimentally that this allows us to learn varied skills in the reset-free setting, and also produces better skills for downstream task learning.

The goal of acquiring primitives for the purpose of enabling efficient learning on downstream tasks is also a central goal of hierarchical reinforcement learning (HRL), which has been a long-standing area of research (Dayan and Hinton, 1993; Wiering and Schmidhuber, 1997; Parr and Russell, 1998; Sutton et al., 1999; Dietterich, 2000). Recent work has investigated deep hierarchical agents (Bacon et al., 2017; Vezhnevets et al., 2017; Florensa et al., 2017; Nachum et al., 2018), building on methods that explicitly design a hierarchy over actions (Parr and Russell, 1998; Dietterich, 2000), or utilize options (Sutton et al., 1999; Precup, 2001). Our work is also motivated by acquiring temporal abstraction through experience, although we seek to do so under a problem setting that is reset-free. Our work can be seen as complementary to prior approaches in HRL, as the skills we acquire can be used by an HRL agent to accelerate downstream learning. We demonstrate this empirically in Section 7.5.

Central to our method is formulating the problem of acquiring diverse skills as an adversarial two player game. Adversarial games have been proposed in RL with the goal of improving exploration (Sukhbaatar et al., 2017a; Lee et al., 2019b), learning goal conditioned policies (Racaniere et al., 2019) and generating an automated curriculum (Tesauro, 1995; Silver et al., 2018; Baker et al., 2019). Our work resembles the reset variant of asymmetric self-play (Sukhbaatar et al., 2017a), though our approach differs in two ways. First, while the reset variant of asymmetric self-play defines "Bob's" learning objective using its ability to reset "Alice's" trajectories, the problem setting is not in fact reset-free as Alice is reset at every episode. Second, our method does not involve setting new goals or use time to completion as a proxy reward, but rather makes use of a task reward to measure which reset states are challenging. Unlike prior methods that formulate an adversarial game around goal reaching (Sukhbaatar et al., 2017a; Racaniere et al., 2019; Florensa et al., 2019), our method does not involve setting or reaching goals, but rather uses a single task to focus the forward policy, providing opportunities for the reset skills to learn varied adversarial perturbations. Unlike methods that adversarially vary the environment (Brant and Stanley, 2017; Wang et al., 2019), our method does not require any additional privileged capability to change the environment parameters. Instead, the reset skills use the same action space as the forward policy to discover states from which the original task is difficult. Finally, unlike all of the previously listed methods, our approach enables learning in a reset-free setting – a setting where, as we show experimentally in Section 7.5, prior methods struggle to learn effectively.

4.2 PRELIMINARIES

Our eventual goal is to devise a reset-free, non-episodic learning framework. We first describe standard episodic RL. An RL problem is defined on a Markov decision process (MDP), represented by the tuple: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p_{\mathbf{s}}, r, \gamma, p_0)$, where $\mathcal{S}$ is a set of continuous states and $\mathcal{A}$ is a set of continuous actions, $p_{\mathbf{s}} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the transition probability density, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, γ is the discount factor and p_0 is the initial state distribution. The γ-discounted return $R(\tau)$ of a trajectory $\tau = (\mathbf{s}_0, \mathbf{a}_0, \ldots \mathbf{s}_{T-1}, \mathbf{a}_{T-1})$ is $\sum_{t=0}^{T-1} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)$. In episodic, finite horizon tasks of length T, the goal is to learn a policy $\pi_\theta : \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ that maximizes the objective $J(\pi_\theta) = \mathbb{E}_{\pi_\theta, p_0, p_{\mathbf{s}}} [R(\tau)]$. Here, θ denotes the parameters of the policy π_θ, which are learned by iteratively sampling episodes, where at the end of each episode, a new initial state is $\mathbf{s}_0$ is sampled from the initial state distribution p_0. While such resets can be easily obtained in simulated tasks (e.g., Atari (Mnih et al., 2013)), this is significantly more challenging in real-world physical tasks, as it requires human intervention in the form of manual resets or the availability of a hard coded reset (Levine et al., 2016a; Vecerik et al., 2017; Rajeswaran et al., 2017b; Yahya et al., 2017; Haarnoja et al., 2018a).

Our method will simultaneously remove the need for manual resets and, at the same time, learn a varied set of skills by resetting the environment in different ways. To learn this set of skills, we build on previously proposed episodic skill learning methods, which we review here. Following prior work (Eysenbach et al., 2018; Sharma et al., 2019), skills can be represented by conditioning the policy on a latent skill index $\mathbf{z}$, which is sampled from $p(\mathbf{z})$ and kept fixed over a temporally extended period. Every distinct value of $\mathbf{z}$ should correspond to a distinct behavior. One objective for achieving this proposed in prior work (Gregor et al., 2016; Eysenbach et al., 2018; Sharma et al., 2019; Pong et al., 2019) is the mutual information (MI) between skills and the states they visit: $\mathcal{I}(\mathbf{s};\mathbf{z}) = \mathcal{H}(\mathbf{s}) - \mathcal{H}(\mathbf{s}|\mathbf{z})$. Maximizing $\mathcal{I}(\mathbf{s};\mathbf{z})$ entails maximizing the state entropy (high state coverage) while minimizing conditional state entropy (high predictability for each $\mathbf{z}$). This is typically combined with minimizing the MI between actions ($\mathbf{a}$) and the state/context ($\mathbf{s}, \mathbf{z}$), so that skills are distinguished by the states they visit rather than the actions they take. The resulting objective (Eysenbach et al., 2018) is:

$$\begin{aligned}\mathcal{I}(\mathbf{s};\mathbf{z}) - \mathcal{I}(\mathbf{a};\mathbf{s},\mathbf{z}) &= \mathbb{E}_\pi \left[\log \frac{p(\mathbf{z}|\mathbf{s})}{p(\mathbf{z})} - \log \frac{\pi(\mathbf{a}|\mathbf{s},\mathbf{z})}{\pi(\mathbf{a})}\right] \\ &\geqslant \mathbb{E}_\pi \left[\log q_\omega(\mathbf{z}|\mathbf{s}) - \log p(\mathbf{z}) - \log \pi(\mathbf{a}|\mathbf{s},\mathbf{z})\right] = \mathcal{G}(\theta, \omega),\end{aligned}$$

where we replace $p(\mathbf{z}|\mathbf{s})$ with an approximate learned discriminator $q_\omega(\mathbf{z}|\mathbf{s})$ with parameters ω to obtain a variational lower bound. We can maximize $\mathcal{G}(\theta, \omega)$ with RL using the pseudo-reward:

$$r_{\text{skill}}(\pi, q_{\omega}) = \log q_{\omega}(\mathbf{z}|\mathbf{s}) - \log p(\mathbf{z}) - \log \pi(\mathbf{a}|\mathbf{s}, \mathbf{z}). \tag{10}$$

The skill learning algorithm alternates between learning the skills with this pseudo-reward and optimizing for a discriminator that is able to discriminate between the skills.

4.3 Learning skills via the reset game

To learn without resets, our approach aims to learn both a *forward* policy $\pi_{\theta}(\mathbf{a}|\mathbf{s})$ and a set of *reset* skills, which we denote $\pi_{\phi}^{\text{reset}}(\mathbf{a}|\mathbf{s}, \mathbf{z})$. We first describe the optimization objectives for each of the policies. Then, we describe how we can balance these two objectives using a game theoretic formulation. Finally, we instantiate this method and present a practical algorithm, where a single goal-oriented forward policy is reset by multiple different reset skills. While learning multiple skills and learning without resets may at first seem like largely unrelated problems, we make the observation that a sufficiently diverse set of skills can serve to alleviate the need for a manual reset, because different policies (i.e., $\pi_{\theta}(\mathbf{a}|\mathbf{s})$ and $\pi_{\phi}^{\text{reset}}(\mathbf{a}|\mathbf{s}, \mathbf{z})$) can reset *each other* into different states. By forcing $\pi_{\theta}(\mathbf{a}|\mathbf{s})$ to succeed from the final states of a range of different reset skills, $\pi_{\theta}(\mathbf{a}|\mathbf{s})$ becomes proficient at performing the task from many states. By forcing the reset skills to all perturb $\pi_{\theta}(\mathbf{a}|\mathbf{s})$ in different ways, the reset skills themselves are forced to differentiate and learn various behaviors, making them suitable for downstream learning of more complex tasks. In this way, the problems of learning without resets and learning diverse skills are a natural fit to be addressed jointly.

Our goal is to devise a method such that our policies have the following properties: (1) executing the reset policy allows the forward policy to learn to solve the task with standard episodic RL (i.e., without an oracle reset to the initial state distribution), (2) through repeated resetting, the *reset* policy is able to discover skills that may be useful for downstream tasks.

4.3.1 *The Reset Game*

We first describe the reset game that represents the core of our method, and then in Section 4.3.2 extend this to incorporate multiple skills by conditioning $\pi_{\phi}^{\text{reset}}$ on $\mathbf{z}$. Our proposed reset game enables reset-free learning via a competitive interaction between

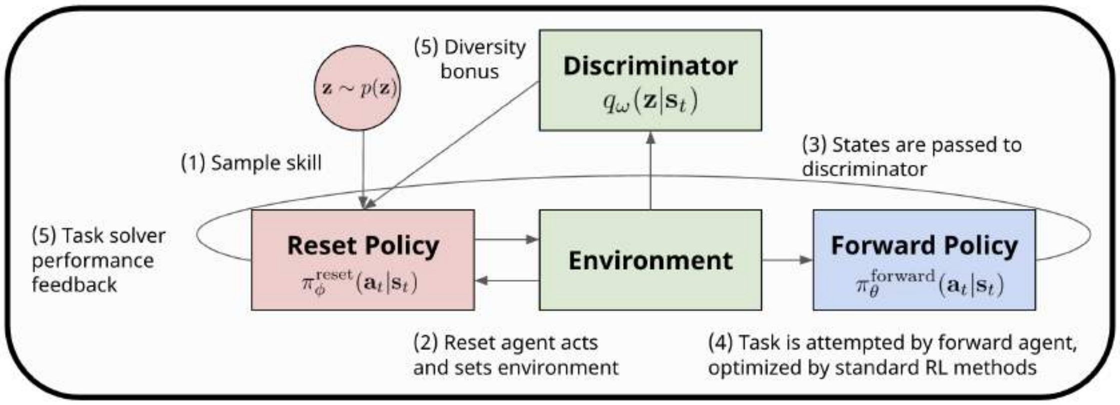

(5) Diversity bonus
$\mathbf{z} \sim p(\mathbf{z})$
Discriminator
$q_\omega(\mathbf{z}|\mathbf{s}_t)$
(1) Sample skill
(3) States are passed to discriminator
(5) Task solver performance feedback
Reset Policy
$\pi_\phi^{\text{reset}}(\mathbf{a}_t|\mathbf{s}_t)$
Environment
Forward Policy
$\pi_\theta^{\text{forward}}(\mathbf{a}_t|\mathbf{s}_t)$
(2) Reset agent acts and sets environment
(4) Task is attempted by forward agent, optimized by standard RL methods

Figure 14: An outline of our approach. Our method first begins by (1) sampling a skill from a prior distribution that is used to condition the reset policy. Next, the reset agent acts in the environment in order to bring the agent to the initial state for the forward policy (2). The states of the reset policy are passed to additionally to a learned discriminator which tries to determine which skill was used to generate the final state (3). Next, the forward policy tries to solve the task (4) and all agents attempt to learn by maximizing their respective objective (5).

$\pi_\theta(\mathbf{a}|\mathbf{s})$ and $\pi_\phi^{\text{reset}}(\mathbf{a}|\mathbf{s})$, specified as follows:

$$\max_{\pi_\theta} \quad \mathcal{J}^{\text{forward}}(\pi_\theta, \pi_\phi^{\text{reset}}), \;\; \max_{\pi_\phi^{\text{reset}}} \quad \mathcal{J}^{\text{reset}}(\pi_\theta, \pi_\phi^{\text{reset}}), \tag{11}$$

where $\mathcal{J}^{\text{forward}}$ and $\mathcal{J}^{\text{reset}}$ are the expected returns, which we define below. Both J^{forward} and J^{reset} depend on both π_θ and π_ϕ^{reset}, since the starting state for each policy is obtained by running the other policy (i.e., $J^{\text{forward}}(\pi_\theta, \pi_\phi^{\text{reset}}) = \mathbb{E}_{\pi_\theta, \pi_\phi^{\text{reset}}, p_s}[R(\tau)]$, and vice-versa).

$\mathcal{J}^{\text{forward}}$ is the task objective, which forces the task policy $\pi_\theta(\mathbf{a}|\mathbf{s})$ to learn to perform the given task. $\mathcal{J}^{\text{reset}}$ is an objective that causes the reset policy $\pi_\phi^{\text{reset}}(\mathbf{a}|\mathbf{s})$ to discover states from which it is difficult for the task policy to succeed at the task, while at the same time promoting diverse and varied resets. We provide specific definitions for these objectives in Section 4.3.2, though a number of different options are possible. Crucially, this is not a zero-sum game. However, due to the fact that the reset policy and task policy must alternate, the reset game has a pre-specified turn order. This form of sequential game is known as a Stackelberg game (Von Stackelberg, 1934), and is closely connected to the field of bi-level optimization (Colson et al., 2007). Unlike general games, these games are known to be solvable (using game theoretic measures) by gradient based learning algorithms (Fiez et al., 2019), even when the game is not zero-sum. Formally, a Stackelberg game is defined as an optimization problem where the "leader" must optimize the following optimization problem:

$$\max_{x_1 \in X_1} \{\, f_1(x_1, x_2) \mid x_2 \in \arg\max_{y \in x_2} f_2(x_1, y) \,\}.$$

The follower optimizes the function $\arg\max_{x_2 \in X_2} f_2(x_1, x_2)$. Taking the reset policy as the leader in our setting results in the following optimization problem:

$$\max_{\pi_\phi^{\text{reset}}} \;\{\, \mathcal{J}^{\text{reset}}(\pi_\theta^*, \pi_\phi^{\text{reset}}) \mid \pi_\theta^* \in \arg\max_{\pi_\theta} \; \mathcal{J}^{\text{forward}}(\pi_\theta, \pi_\phi^{\text{reset}}) \,\}. \tag{12}$$

Based on this connection, we can devise a gradient-based algorithm to solve the reset game in Equation 11, so long as the reset policy and task policy learn at different rates (e.g., with the task policy learning faster). We present such an algorithm in Section 4.3.3, but first we describe how the reset policy can discover diverse skills during the reset.

4.3.2 *Mining Diverse Skills from Diverse Resets*

Recall that our goal is to allow the forward policy to learn to optimize its task reward r, as if a reset mechanism were present, and for our reset mechanism to provide challenging and diverse resets. Towards our first goal, we propose to optimize the forward RL policy from the initial state distribution implied by executing π_ϕ^{reset} for T_{reset} time steps. Towards our second goal, we propose to use the skill learning reward r_{skill} defined in Equation 10 to simultaneously acquire a repertoire of reset skills that are forced to diversify. The underlying mutual information objective encourages high coverage of initial states, but to ensure that these states are challenging for π_θ, we add the negative return of π_θ to the objective for each reset skill. Putting these terms in the game defined in Equation 12 results in the following optimization problem:

$$\max_{\phi} \; \underbrace{\mathbb{E}_{\mathbf{s}_t{}', \mathbf{a}_t{}' \sim \pi_\phi^{\text{reset}}} \left[\sum_{t=0}^{T_{reset}-1} \gamma^t r_{\text{skill}}(\mathbf{a}_t{}', \mathbf{s}_t{}') - \lambda \, \mathbb{E}_{\pi_{\theta^*}} \left[\sum_{t=0}^{T-1} \gamma^t r(\mathbf{a}_t, \mathbf{s}_t) \right] \right]}_{\mathcal{J}^{\text{reset}}(\pi_\theta, \pi_\phi^{\text{reset}})} \tag{13}$$

$$\text{such that } \theta^* = \arg\max_{\theta} \; \underbrace{\mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{T-1} \gamma^t r(\mathbf{a}_t, \mathbf{s}_t) \right]}_{\mathcal{J}^{\text{forward}}(\pi_\theta, \pi_\phi^{\text{reset}})}. \tag{14}$$

Algorithm 4 Learning Skillful Resets (LSR)

```
1: **Input:** Environment
2: **Initialize:** policy $\pi_\theta$, reset policy $\pi_\phi^{\text{reset}}$, discriminator $q_\omega(\mathbf{s}_t|\mathbf{a}_t)$, prior $p(z)$
3: **for** N iterations **do**
4:     Sample skill $\mathbf{z} \sim p(\mathbf{z})$
5:     # Set environment
6:     **for** $t \leftarrow 0 \dots T_{\text{reset}-1}$ **do**
7:         Sample $\mathbf{a}_t \sim \pi_\phi^{\text{reset}}(\mathbf{a}_t|\mathbf{s}_t, \mathbf{z})$
8:         Step env $\mathbf{s}_{t+1} \sim p_{\mathbf{s}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, compute reward for reset controller $r_t^{\text{reset}} = q_\omega(\mathbf{s}_t|\mathbf{a}_t)$
9:     # Solve task
10:    **for** $t \leftarrow 0 \dots T-1$ **do**
11:        Sample $\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$
12:        Step env $\mathbf{s}_{t+1} \sim p_{\mathbf{s}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, obtain environment reward $r_t$
13:    Update reset policy's final reward $r_T^{\text{reset}} = -\sum_{t=0}^{T} \gamma^t r_t(\mathbf{a}_t, \mathbf{s}_t)$
14:    Update $\pi_\phi^{\text{reset}}$, $\pi_\theta$ to maximize respective return using SAC
15:    Update discriminator $q_\omega$ using Adam.
```

The hyperparameter λ controls the relative importance of the two terms of $\mathcal{J}^{\text{reset}}(\pi_\theta, \pi_\phi^{\text{reset}})$. Intuitively, setting $\lambda = 0$ reduces the objective of the reset controller to prior skill learning methods (Eysenbach et al., 2018), while $\lambda > 0$ requires the reset controller to reach more challenging terminal states. As the task policy and reset skills improve, they provide a curriculum for one another, with the task policy pushing the skills to reach more distant states, and the skills pushing the policy to be effective from a larger range of initializations. As we demonstrate in our experiments, this results in more complex and varied skills, as well as faster reset-free learning for π_θ. To reflect the synthesis of resets and skill learning, we refer to our specific instantiation of the reset game as **Learning Skills from Resets** (LSR).

Running this reset game produces an effective forward policy $\pi_\theta(\mathbf{a}|\mathbf{s})$, as well as a set of skills defined by $\pi_\phi^{\text{reset}}(\mathbf{a}|\mathbf{s}, \mathbf{z})$. The latter can be used to solve more complex downstream tasks in a hierarchical RL framework, using a higher-level policy $\pi^{\text{hi}}(\mathbf{z}|\mathbf{s})$ to command the skill $\mathbf{z}$ that should be executed at each (higher-level) time step. We demonstrate this capability in our experiments.

4.3.3 *Algorithm Summary: Optimizing the Reset Game*

We could optimize the task policy and reset skills as per Equation 13 via a bi-level optimization, where the parameters of π_ϕ^{reset} are held fixed while π_θ is optimized to convergence. However, in an RL setting, this would require an excessive number of samples. A substantially more efficient method alternates gradient steps on both objectives, with different learning rates. Since the reset game corresponds to a Stackelberg game, gradients-based learning algorithms with respect to these objectives exist that have finite time high probability bounds for local convergence in the case of non zero-sum objectives (Fiez et al., 2019). Optimizing such a problem can be done approximately by using a two-timescale algorithm, where the leader is optimized at a smaller learning rate (Fiez et al., 2019). This intuitively results in a reset controller that changes "slower" relative to the forward policy. In our implementation, we approximately optimize $\pi_\theta(\mathbf{a}|\mathbf{s})$ and $\pi_\phi^{\text{reset}}(\mathbf{a}|\mathbf{s}, \mathbf{z})$ using soft actor-critic (SAC) (Haarnoja et al., 2018c).

We outline the pseudo-code of our approach in Algorithm 4. At each iteration, we sample a reset skill $\mathbf{z} \sim p(\mathbf{z})$, execute that skill by following $\pi_\phi^{\text{reset}}(\mathbf{a}_t|\mathbf{s}_t, \mathbf{z})$ to reset to a challenging initial state, then execute $\pi_\theta(\mathbf{a}|\mathbf{s})$ to attempt the task. The experience collected during these trials is then used to update the corresponding policies.

4.4 EXPERIMENTS

In this section we aim to experimentally answer the following questions: (1) How does our approach compare to prior methods in the reset free domain? (2) How do resets and state representation choice effect the skills we learn? (3) How do the skills learned by our approach compare to prior work? Specifically, can we learn better hierarchical controllers using the primitives learned by our approach (LSR)? We first describe our experimental setup, evaluation metrics, and the prior methods to which we will compare. We leave detailed discussion of hyperparameters and environment parameters to the Appendix C.

Experimental setup. To study reset-free learning, we use the three-fingered hand repositioning task proposed by Zhu et al. (2020a), where the hand must position an object in the center of a bin, starting from any position. This is the most challenging domain considered by Zhu et al. (2020a) for reset-free learning. We refer to this environment as the `DClaw-ManipulateFreeObject` environment. We follow the evaluation protocol of Zhu et al. (2020a), using an evaluation metric that measures the final position and orientation of the object compared to the goal position evaluated from a set of initial positions that

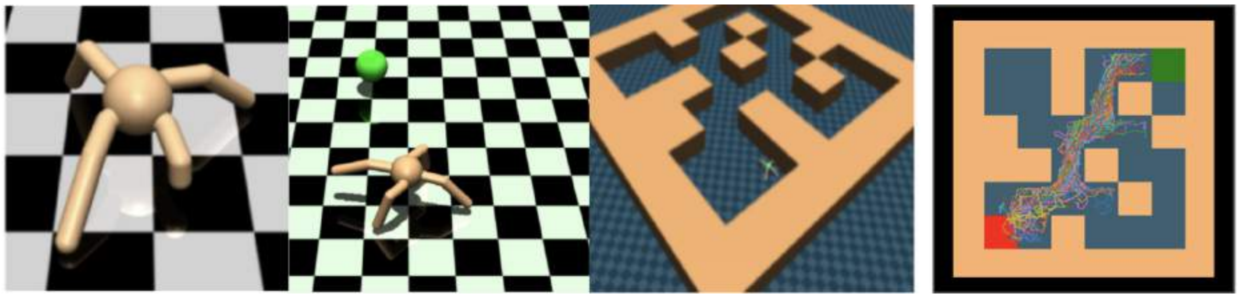


Figure 15: Diagram of the hierarchical locomotion tasks considered in this work. The agent must first acquire locomotion skills in a reset game (left), where the task policy must learn to walk to the origin of the workspace. The skills are subsequently used as the action space for a hierarchical policy, which must learn complex hierarchical navigation tasks (second and third image). The far-right image shows the paths taken by a hierarchical policy using our learned reset skills.

is consistent across all methods. To evaluate our second experimental hypothesis, we evaluate skills acquired with reset-free learning for an ant locomotion task on two hierarchical navigation problems, shown in Figure 15. In these experiments, the agent must first learn a set of skills (without resets) to control an `Ant` quadrupedal robot to walk to the center of the workspace (far left), and subsequently transfer those skills to a waypoint navigation task and a maze traversal environment, which we refer to as `Ant-Waypoint` and `Ant-MediumMaze` (center, right plot respectively). For all methods, we report a return, which we normalize using the return of an agent that successfully solves the task and a random agent.

Comparisons and baselines. We compare our approach to prior methods, which are trained either with or without resets. In the `DClaw` environments, we compare to two prior methods that learn reset controllers: leave no trace (LNT) Eysenbach et al., 2017, and the R3L perturbation controller (R3L) Zhu et al., 2020a. In order to fairly compare all prior methods, we use the same underlying RL algorithm, soft actor-critic (SAC) (Haarnoja et al., 2018c), across all methods, and use the same exploration bonus used in R3L Zhu et al., 2020a, random network distillation Burda et al., 2018, which is added to the reward for all methods.

We also compare to prior skill learning methods on the `Ant` task: Diversity is All You Need (DIAYN) (Eysenbach et al., 2018) and Dynamics-Aware Discovery of Skills (DADS) (Sharma et al., 2019). DADS and DIAYN are unsupervised skill learning procedures that aim to learn a diverse set of skills in an environment. For this domain, prior work has differed in the choice of state representation, either using the full state Eysenbach et al., 2018, which is a generally more challenging setting, or a reduced (x, y)

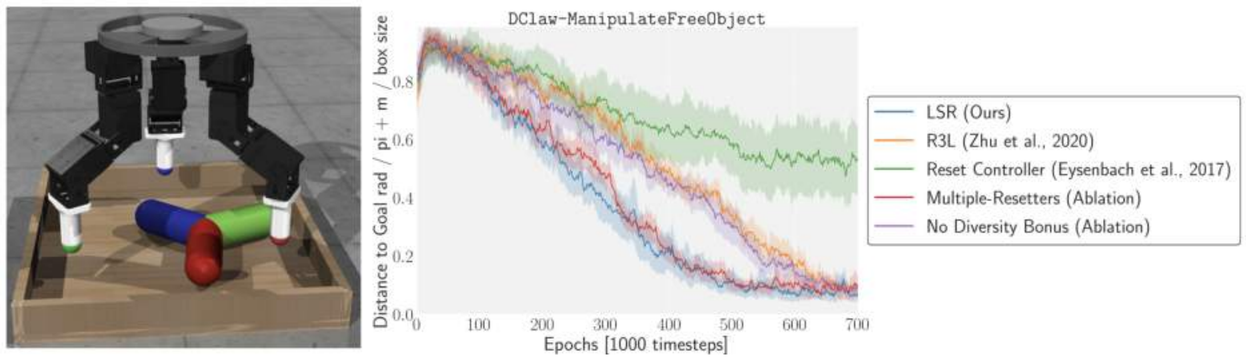


Figure 16: Reset-free learning comparison (lower is better). The error bars show 95% bootstrap confidence intervals for average performance. We average each method over 4 seeds. Our method (blue) outperforms prior methods (orange, green). Compared to R3L Zhu et al., 2020a (orange), our approach converges ~28.6% faster. Our ablations show that the most important aspect of our approach is having multiple resetters. This indicates that reset state convergence is crucial to good performance, in contrast to prior methods Eysenbach et al., 2017 that learn a explicit reset policy.

coordinate state representation. For our experiments, in order to understand the effect of resets and state representation choice, we experiment with both the full-state representation and reduced (x, y) coordinate state space, in addition to running all methods with resets or reset-free. In the case of DADS Sharma et al., 2019, which uses continuous skills, we discretize the skill space to allow us to use the same hierarchical learner.

Reset-free learning results. The results in Figure 16 show that our approach substantially improves reset-free performance. In contrast to prior approaches (orange and green), our method converges approximately 200 iterations faster on the most challenging task (DClaw-ManipulateFreeObject), which is a 28.6% improvement. We hypothesize that is due to the improved state coverage of our approach, as an explicit reset policy (green curve) (Eysenbach et al., 2017) performs much worse. This is supported by our ablations, which show that having "Multiple-Resetters" (red), where we remove the diversity bonus, accounts for much of the method's improvements.

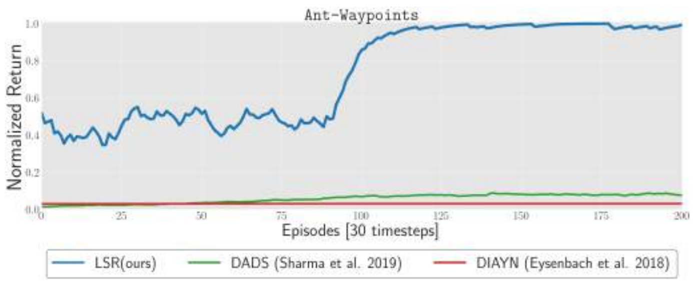


Figure 18: Performance on the `Ant-Waypoints` environment compared to DADS and DIAYN. DADS and DIAYN makes small progress on this simpler domain (orange), but our method (blue) is still able to successfully navigate between the waypoints more quickly.

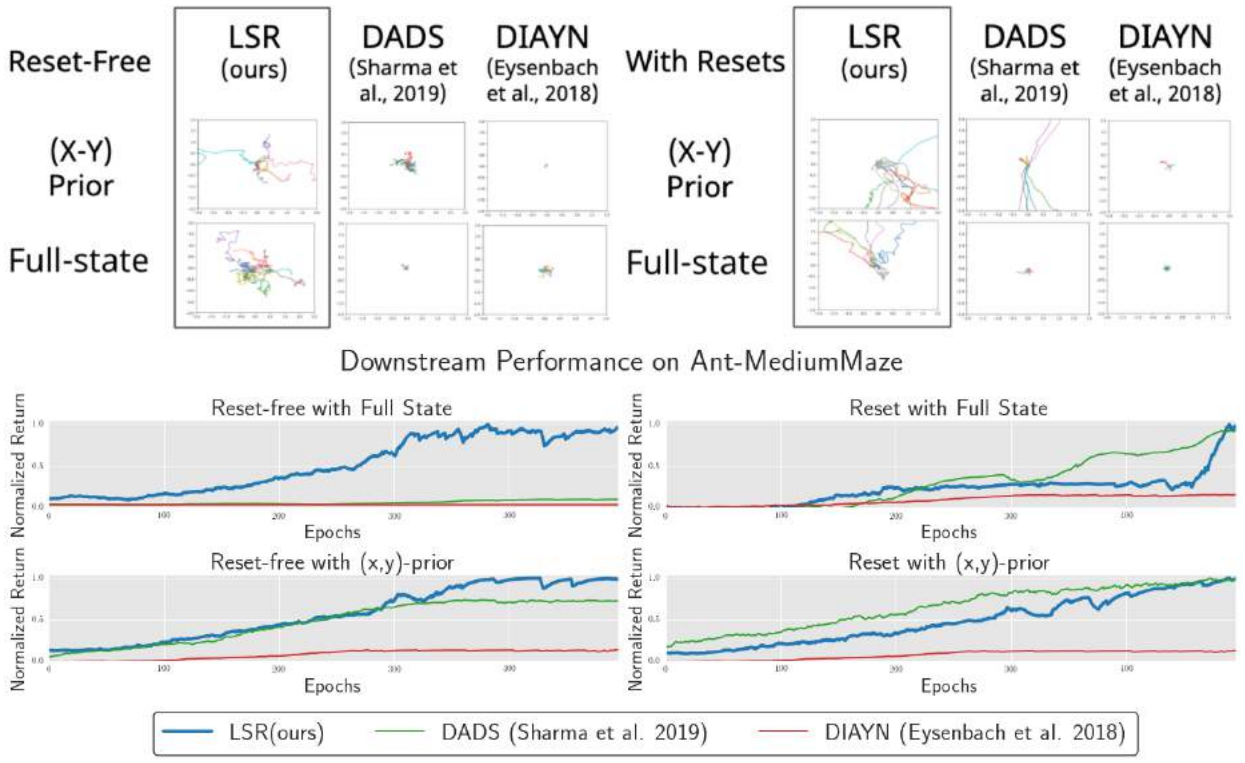


Figure 17: A visualization of the learned skills (top) and performance on a hierarchical downstream locomotion task (bottom) under different state representation choices and reset availability. We visualize the (x, y) trajectories of learned skills. In comparison to prior methods, LSR is able to learn skills that are able to robustly move the ant much further from the origin compared to prior approaches which either need the (x, y)-prior or resets. This allows us to substantially improve performance in a downstream hierarchical locomotion task (bottom, top-left), where a meta-controller using only the skills is able to navigate a maze much more effectively. .

Using a single adversary (purple-curve), where we run our approach with a single skill, performs comparably to R3L, and substantially worse than our method. Note that prior techniques that have formulated adversarial games, such as asymmetric self play (ASP) (Sukhbaatar et al., 2017a), generally use a single adversary.[1]

Choice of state representation and the effect of resets. In the `Ant` domain, we find that our approach allows us to robustly learn skills across reset settings and state representation choice. We visualize the (x, y) trajectories of the learned reset skills qualitatively in

1 While ASP shares many properties with our method, such as the use of a game formulation, it was infeasible to compare on these tasks due to differences in assumptions: ASP does not consider reset-free learning, does not aim to optimize a given task, and requires goal-reaching policy formulations for both players. However, we can view the "Single Adversary" baseline as the closest to ASP in our experiments.

Figure 46 (top), along with skills learned by prior methods with and without resets (Eysenbach et al., 2018; Sharma et al., 2019). Both prior methods (DIAYN and DADS) learn short directional walking skills with resets (top, right two columns), similar to those reported in prior work (Eysenbach et al., 2018; Sharma et al., 2019), but generally fail to learn meaningful skills in the reset-free setting. Performance is generally improved by using the (x, y) prior as the state representation, but this is not robust in the reset-free setting for prior methods. In contrast, the skills learned by our method are capable of moving the ant much further than prior methods that learn in *either* the reset-free or reset-based setting, with both the full-state and (x, y) representation, as illustrated in the figure.

Skill learning and hierarchical control results. Finally, we compare our approach to prior methods by evaluating the quantitative performance on downstream hierarchical tasks. we see that the skills produced by our approach improves performance in the simpler `Ant-WayPoints` navigation domain Fig. 18, and these improvements are much more pronounced in the more challenging `Ant-MediumMaze` task, where our approach leads to substantially more effective downstream learning, as shown in Fig. 46 (right), compared to prior methods. We normalize each curve in Fig. 46 by the return of the best performing policy. A visualization of the path taken through the maze by our hierarchical policy in the reset-free, full state representation setting is shown in Fig. 15 (far right). We provide a similar analysis on the `Ant-Waypoints` task in Appendix. D.2.

4.5 DISCUSSION, LIMITATIONS AND FUTURE WORK

We proposed a method that simultaneously addresses two challenges faced by real-world RL systems: learning continually without manually provided resets, and acquiring diverse skills for solving long-horizon downstream tasks. While these two problems may at first appear unrelated, we show that learning diverse skills actually enables more effective reset-free learning, while learning the skills in an adversarial manner with a task policy actually makes the skills themselves more effective, and therefore better suited for downstream long-horizon problems. Our work assumed environments that are reversible, which is a strong assumption assumption in the real world. We also assumed a defined forward task, whereas a fully autonomous real world agent would need to have the ability to verify its own task success. A particularly exciting direction for future work would be to extend the reset game to learn a zero or few-shot success classifier, or design a system that explore conservatively and request limited human intervention when encountering a non-communicating state.

5

MULTI-TASK RESET-FREE LEARNING FOR DEXTEROUS MANIPULATION

In Chapter 4, we discussed how learning multiple skills can be helpful for confronting the challenge of non-episodic learning in the real world. In this chapter, we focus specifically on real world learning of dexterous manipulation skills, which presents a particularly clear lens on the reset-free learning problem and the utility of learning skill learning. For instance, a dexterous hand performing in-hand manipulation, as shown in Figure 19 (right), must delicately balance the forces on the object to keep it in position. Early on in training, the policy will frequently drop the object, necessitating a particularly complex reset procedure. Prior work has addressed this manually by having a human involved in the training process Ploeger et al., 2020; Kumar et al., 2016b; Ghadirzadeh et al., 2017a, instrumenting a reset in the environment Zhu et al., 2019; Chebotar et al., 2016, or even by programming a separate robot to place the object back in the hand Nagabandi et al., 2020. Though some prior techniques have sought to learn some form of a "reset" controller Eysenbach et al., 2017; Ahn et al., 2020; Agrawal et al., 2016; Zhu et al., 2020b; Sukhbaatar et al., 2017b; Richter and Roy, 2017, none of these are able to successfully scale to solve a complex dexterous manipulation problem without hand-designed reset systems due to the challenge of learning robust reset behaviors.

However, general-purpose robots deployed in real-world settings will likely be tasked with performing many different behaviors. While multi-task learning algorithms in these settings have typically been studied in the context of improving sample efficiency and generalization Teh et al., 2017; Rusu et al., 2016a; Parisotto et al., 2016; Yu et al., 2020a; Sener and Koltun, 2018; Yu et al., 2019, in this work we make the observation that multi-task algorithms naturally lend themselves to the reset-free learning problem. We hypothesize that the reset-free RL problem can be addressed by reformulating it as a multi-task problem, and appropriately sequencing the tasks commanded and learned during online reinforcement learning. As outlined in Figure 24, solving a collection of tasks simultane-

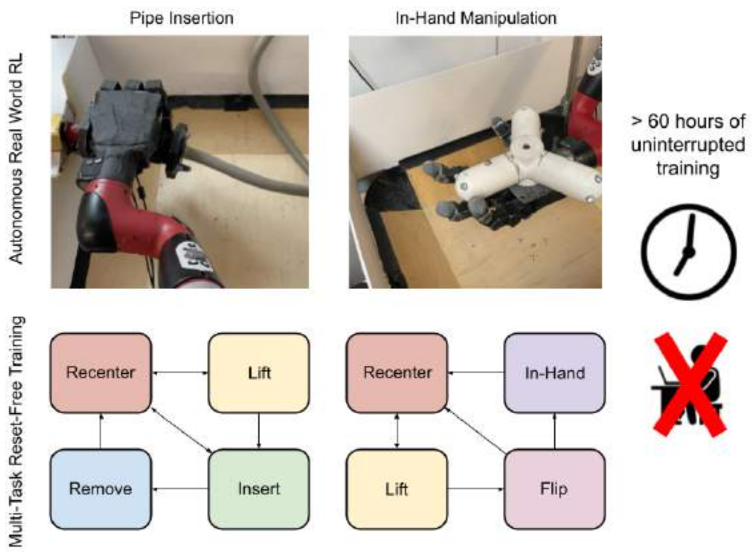


Figure 19: Reset-free learning of dexterous manipulation behaviors by leveraging multi-task learning. When multiple tasks are learned together, different tasks can serve to reset each other, allowing for uninterrupted continuous learning of all of the tasks. This allows for the learning of dexterous manipulation tasks like in-hand manipulation and pipe insertion with a 4-fingered robotic hand, without any human intervention, with over 60 hours of uninterrupted training

ously presents the possibility of using some tasks as a reset for others, thereby removing the need of explicit per task resets. For instance, if we consider the problem of learning a variety of dexterous hand manipulation behaviors, such as in-hand reorientation, then learning and executing behaviors such as recenter and pickup can naturally reset the other tasks in the event of failure (as we describe in Section 6.4 and Figure 24). We show that by learning multiple different tasks simultaneously and appropriately sequencing behavior across different tasks, we can learn *all* of the tasks without episodic resets required at all. This allows us to effectively learn a "network" of reset behaviors, each of which is easier than learning a complete reset controller, but together can execute and learn more complex behavior.

The main contribution of this work is to propose a learning system that can learn dexterous manipulation behaviors without the need for episodic resets. We do so by leveraging the paradigm of multi-task reinforcement learning to make the reset free problem less challenging. The system accepts a diverse set of tasks that are to be learned, and then trains reset-free, leveraging progress in some tasks to provide resets for other tasks. To validate this algorithm for reset-free robotic learning, we perform both simulated and hardware experiments on a dexterous manipulation system with a four fingered anthropomorphic robotic hand. To our knowledge, these results demonstrate the first instance of a combined hand-arm system learning dexterous in-hand manipulation with deep RL entirely in the real world with minimal human intervention during training, simultaneously acquiring both the in-hand manipulation skill and the skills needed to retry the task. We also show the ability of this system to learn other dexterous manipulation behaviors like pipe insertion via uninterrupted real world training, as well as several tasks in simulation.

5.1 RELATED WORK

RL algorithms have been applied to a variety of robotics problems in simulation James et al., 2019; Rajeswaran et al., 2017a; Heess et al., 2017; OpenAI et al., 2018, and have also seen application to real-world problems, such as locomotion Ha et al., 2020; Peng et al., 2020; Calandra et al., 2016, grasping Kalashnikov et al., 2018; Levine et al., 2016b; Baier-Löwenstein and Zhang, 2007; Wu et al., 2019, manipulation of articulated objects Nemec et al., 2017; Urakami et al., 2019; Chebotar et al., 2016; Gu et al., 2016; Nair et al., 2018, and even dexterous manipulation Hoof et al., 2015a; Kumar et al., 2016b. Several prior works Okamura et al., 2000 have shown how to acquire dexterous manipulation behaviors with optimization Furukawa et al., 2006; Bai and Liu, 2014; Mordatch et al., 2012; Kumar et al., 2014a; Yamane et al., 2004, reinforcement learning in simulation Rajeswaran et al., 2017a; Mandikal and Grauman, 2020a; Jain et al., 2019, and even in the real world OpenAI, 2018; Ploeger et al., 2020; Hoof et al., 2015b; Nagabandi et al., 2020; Ahn et al., 2020; Zhu et al., 2019; Kumar et al., 2016b; Gupta et al., 2016a; Choi et al., 2018; Kumar et al., 2016a. These techniques have leaned on highly instrumented setups to provide episodic resets and rewards. For instance, prior work uses a scripted second arm Nagabandi et al., 2020 or separate servo motors Zhu et al., 2019 to perform resets. Contrary to these, our work focuses on removing the need for explicit environment resets, by leveraging multi-task learning.

Our work is certainly not the first to consider the problem of reset-free RL Montgomery et al., 2016. Eysenbach et al., 2017 proposes a scheme that interleaves attempts at the task with episodes from an explicitly learned reset controller, trained to reach the initial state. Building on this work, Zhu et al., 2020b shows how to learn simple dexterous manipulation tasks without instrumentation using a perturbation controller exploring for novelty instead of a reset controller. Han et al., 2015; Smith et al., 2019 demonstrate learning of multi-stage tasks by progressively sequencing a chain of forward and backward controllers. Perhaps the most closely related work to ours algorithmically is the framework proposed in Ha et al., 2020, where the agent learns locomotion by leveraging multi-task behavior. However, this work studies tasks with cyclic dependencies specifically tailored towards the locomotion domain. Our work shows that having a variety of tasks and learning them all together via multi-task RL can allow solutions to challenging reset free problems in dexterous manipulation domains.

Our work builds on the framework of multi-task learning Teh et al., 2017; Rusu et al., 2016a; Parisotto et al., 2016; Yu et al., 2020a; Sener and Koltun, 2018; Yu et al., 2019; Ruder, 2017; Yang et al., 2020 which have been leveraged to learn a collection of behaviors, improve on generalization as well as sample efficiency, and have even applied to real world robotics problems Riedmiller et al., 2018; Wulfmeier et al., 2019; Deisenroth et al., 2014. In this work, we take a different view on multi-task RL as a means to solve reset-free learning problems.

5.2 PRELIMINARIES

We build on the framework of Markov decision processes for reinforcement learning. We refer the reader to Sutton and Barto, 2018a for a more detailed overview. RL algorithms consider the problem of learning a policy $\pi(a|s)$ such that the expected sum of rewards $R(s_t, a_t)$ obtained under such a policy is maximized when starting from an initial state distribution μ_0 and dynamics $\mathcal{P}(s_{t+1}|s_t, a_t)$. This objective is given by:

$$J(\pi) = \mathbb{E}_{\substack{s_0 \sim \mu_0 \\ a_t \sim \pi(a_t|s_t) \\ s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t)}} \left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t) \right] \tag{15}$$

While many algorithms exist to optimize this objective, in this work we build on the framework of actor-critic algorithms Konda and Tsitsiklis, 1999. Although we build on actor critic framework, we emphasize that our framework can be effectively used with many standard reinforcement learning algorithms with minimal modifications.

As we note in the following section, we address the reset-free RL problem via multi-task RL. Multi-task RL attempts to learn multiple tasks simultaneously. Under this setting, each of K tasks involves a separate reward function R_i, different initial state distribution μ_0^i and potentially different optimal policy π_i. Given a distribution over the tasks

$p(i)$, the multi-task problem can then be described as

$$J(\pi_0, \ldots, \pi_{K-1}) = \mathbb{E}_{i \sim p(i)} \mathbb{E}_{\substack{s_0 \sim \mu_0^i \\ a_t \sim \pi_i(s_t|a_t) \\ s_{t+1} \sim \mathcal{P}(s_t, a_t)}} \left[\sum_t \gamma^t R_i(s_t, a_t) \right] \tag{16}$$

In the following section, we will discuss how viewing reset-free learning through the lens of multi-task learning can naturally address the challenges in reset-free RL.

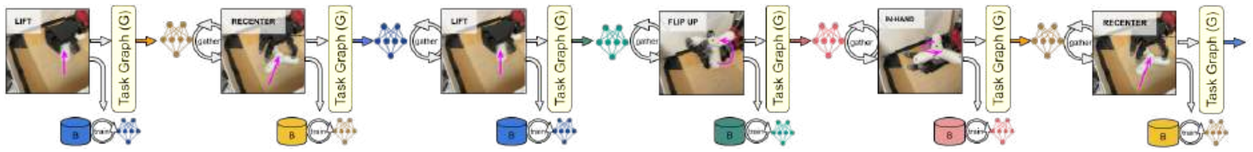


Figure 20: Depiction of some steps of reset-free training for the in-hand manipulation task family on hardware. Reset-free training uses the task graph to choose which policy to execute and train at every step. For executions that are not successful (e.g., the pickup in step 1), other tasks (recenter in step 2) serve to provide a reset so that pickup can be attempted again. Once pickup is successful, the next task (flip up) can be attempted. If the flip-up policy is successful, then the in-hand reorientation task can be attempted and if this drops the object then the re-centering task is activated to continue training.

5.3 LEARNING DEXTEROUS MANIPULATION BEHAVIORS RESET-FREE VIA MULTI-TASK RL

One of the main advantages of dexterous robots is their ability to perform a wide range of *different* tasks. Indeed, we might imagine that a real-world dexterous robotic system deployed in a realistic environment, such as a home or office, would likely need to perform a repertoire of different behaviors, rather than just a single skill. While this may at first seem like it would only make the problem of learning without resets more difficult, the key observation we make in this work is that the multi-task setting can actually facilitate reset-free learning without manually provided instrumentation. When a large number of diverse tasks are being learned simultaneously, some tasks can naturally serve as resets for other tasks during learning. Learning each of the tasks individually without resets is made easier by appropriately learning and sequencing together other tasks in the right order. By doing so, we can replace the simple forward, reset behavior dichotomy with a more natural "network" of multiple tasks that can perform complex reset behaviors between each other.

Let us ground this intuition in a concrete example. Given a dexterous table-top manipulation task, shown in Fig 24 and Fig 20 , our reset-free RL procedure might look like this: let us say the robot starts with the object in the palm and is trying to learn how to manipulate it in-hand so that it is oriented in a particular direction (*in-hand reorient*). While doing so, it may end up dropping the object. When learning with resets, a person would need to pick up the object and place it back in the hand to continue training. However, since we would like the robot to learn without such manual interventions, the robot itself needs to retrieve the object and resume practicing. To do so, the robot must first *re-center* and the object so that it is suitable for grasping, and then actually *lift* and the *flip up* so it's in the palm to resume practicing. In case any of the intermediate tasks (say lifting the object) fails, the recenter task can be deployed to attempt picking up again, practicing these tasks themselves in the process. Appropriately sequencing the execution and learning of different tasks, allows for the autonomous practicing of in-hand manipulation behavior, without requiring any human or instrumented resets.

5.3.1 *Algorithm Description*

In this work, we directly leverage this insight to build a dexterous robotic system that learns in the absence of resets. We assume that we are provided with K different tasks that need to be learned together. These tasks each represent some distinct capability of the agent. As described above, in the dexterous manipulation domain the tasks might involve re-centering, picking up the object, and reorienting an object in-hand. Each of these K different tasks is provided with its own reward function $R_i(s_t, a_t)$, and at test-time is evaluated against its distinct initial state distribution μ_0^i.

Our proposed learning system, which we call Multi-Task learning for Reset-Free RL (MTRF), attempts to jointly learn K different policies π_i, one for each of the defined tasks, by leveraging off-policy RL and continuous data collection in the environment. The system is initialized by randomly sampling a task and state s_0 sampled from the task's initial state distribution. The robot collects a continuous stream of data *without any subsequent resets* in the environment by sequencing the K policies according to a meta-controller (referred to as a "task-graph") $G(s) : S \to \{0, 1, \dots, K-1\}$. Given the current state of the environment and the learning process, the task-graph makes a decision once every T time steps on which of the tasks should be executed and trained for the *next* T time steps. This task-graph decides what order the tasks should be learned and which of the policies should be used for data collection. The learning proceeds by iteratively collecting data with a policy π_i chosen by the task-graph for T time steps, after which the collected data is saved to a task-specific replay buffer $\mathcal{B}_i$ and the task-graph is queried again for which task to execute next, and the whole process repeats.

We assume that, for each task, successful outcomes for tasks that lead into that task according to the task graph (i.e., all incoming edges) will result in valid initial states for that task. This assumption is reasonable: intuitively, it states that an edge from task A to task B implies that successful outcomes of task A are valid initial states for task B. This means that, if task B is triggered after task A, it will learn to succeed from these valid initial states under μ_0^B. While this does not always guarantee that the downstream controller for task B will see *all* of the initial states from μ_0^B, since the upstream controller is not explicitly optimizing for coverage, in practice we find that this still performs very well. However, we expect that it would also be straightforward to introduce coverage into this method by utilizing state marginal matching methods Lee et al., 2019a. We leave this for future work.

The individual policies can continue to be trained by leveraging the data collected in their individual replay buffers $\mathcal{B}_i$ via off-policy RL. As individual tasks become more and more successful, they can start to serve as effective resets for other tasks, forming a natural curriculum. The proposed framework is general and capable of learning a diverse collection of tasks reset-free when provided with a task graph that leverages the diversity of tasks. This leaves open the question of how to actually define the task-graph G to effectively sequence tasks. In this work, we assume that a task-graph defining the various tasks and the associated transitions is provided to the agent by the algorithm designer. In practice, providing such a graph is simple for a human user, although it could in principle be learned from experiential data. We leave this extension for future work.

Interestingly, many other reset-free learning methods Eysenbach et al., 2017; Han et al., 2015; Smith et al., 2019 can be seen as special cases of the framework we have described above. In our experiments we incorporate one of the tasks as a "perturbation" task. While prior work considered doing this with a single forward controller Zhu et al., 2020b, we show that this type of perturbation can generally be applied by simply viewing it as another task. We incorporate this perturbation task in our instantiation of the algorithm, but we do not show it in the task graph figures for simplicity.

5.3.2 *Practical Instantiation*

To instantiate the algorithmic idea described above as a deep reinforcement learning framework that is capable of solving dexterous manipulation tasks without resets, we can build on the framework of actor-critic algorithms. We learn separate policies π_i for each of the K provided tasks, with separate critics Q_i and replay buffers $\mathcal{B}_i$ for each of the tasks. Each of the policies π_i is a deep neural network Gaussian policy with parameters θ_i, which is trained using a standard actor-critic algorithm, such as soft actor-critic Haarnoja et al., 2018c, using data sampled from its own replay buffer $\mathcal{B}_i$. The task graph G is represented as a user-provided state machine, as shown in Fig 24, and is queried every T steps to determine which task policy π_i to execute and update next. Training proceeds by starting execution from a particular state s_0 in the environment, querying the task-graph G to determine which policy $i = G(s_0)$ to execute, and then collecting T time-steps of data using the policy π_i, transitioning the environment to a new state s_T (Fig 24). The task-graph is then queried again and the process is repeated until all the tasks are learned.

Algorithm 5 MTRF

```
1: Given: K tasks with rewards $R_i(s_t, a_t)$, along with a task graph mapping states to a
   task index $G(s) : S \to \{0, 1, \ldots, K-1\}$
2: Let $\hat{i}$ represent the task index associated with the forward task that is being learned.
3: Initialize $\pi_i$, $Q_i$, $\mathcal{B}_i$ $\forall i \in \{0, 1, \ldots, K-1\}$
4: Initialize the environment in task $\hat{i}$ with initial state $s_{\hat{i}} \sim \mu_{\hat{i}}(s_{\hat{i}})$
5: for iteration $n = 1, 2, \ldots$ do
6:     Obtain current task $i$ to execute by querying task graph at the current environ-
       ment state $i = G(s_{curr})$
7:     for iteration $j = 1, 2, \ldots, T$ do
8:         Execute $\pi_i$ in environment, receiving task-specific rewards $R_i$ storing data in
           the buffer $\mathcal{B}_i$
9:         Train the current task's policy and value functions $\pi_i$, $Q_i$ by sampling a
           batch from the replay buffer containing this task's experience $\mathcal{B}_i$, according to SAC
           Haarnoja et al., 2018c.
```

5.4 TASK AND SYSTEM SETUP

To study MTRF in the context of challenging robotic tasks, such as dexterous manipulation, we designed an anthropomorphic manipulation platform in both simulation and hardware. Our system (Fig 23) consists of a 22-DoF anthropomorphic hand-arm system. We use a self-designed and manufactured four-fingered, 16 DoF robot hand called the *D'Hand*, mounted on a 6 DoF Sawyer robotic arm to allow it to operate in an extended workspace in a table-top setting. We built this hardware to be particularly amenable to our problem setting due it's robustness and ease of long term operation. The *D'Hand* can operate for upwards of 100 hours in contact rich tasks without any breakages, whereas previous hand based systems are much more fragile. Given the modular nature of the hand, even if a particular joint malfunctions, it is quick to repair and continue training. In our experimental evaluation, we use two different sets of dexterous manipulation tasks in simulation and two different sets of tasks in the real world. Details can be found in Appendix A.

5.4.1 *Simulation Domains*

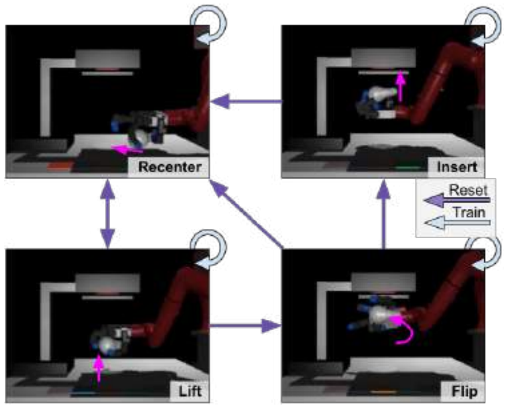


Figure 21: Tasks and transitions for lightbulb insertion in simulation. The goal is to recenter a lightbulb, lift it, flip it over, and then insert it into a lamp.

Lightbulb insertion tasks. The first family of tasks involves inserting a lightbulb into a lamp in simulation with the dexterous hand-arm system. The tasks consist of centering the object on the table, pickup, in-hand reorientation, and insertion into the lamp. The multi-task transition task graph is shown in Fig 21. These tasks all involve coordinated finger and arm motion and require precise movement to insert the lightbulb.
Basketball tasks. The second family of tasks involves dunking a basketball into a hoop. This consists of repositioning the ball, picking it up, positioning the hand over the basket, and dunking the ball. This task has a natural cyclic nature, and allows tasks to reset each other as shown in Fig 22, while requiring fine-grained behavior to manipulate the ball midair.

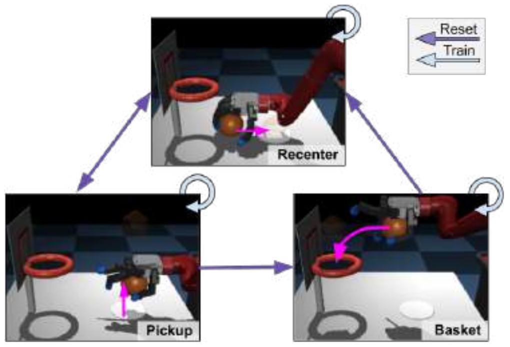


Figure 22: Tasks and transitions for basketball domain in simulation. The goal here is to reposition a basketball object, pick it up and then dunk it in a hoop.

5.4.2 *Hardware Tasks*

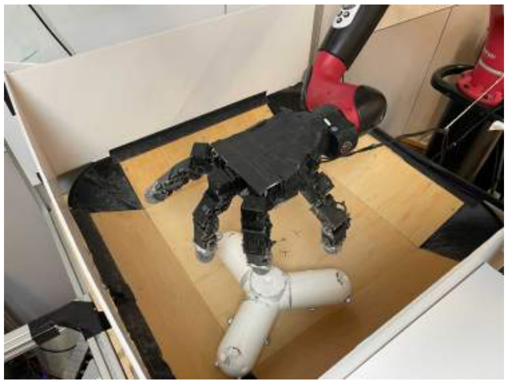


Figure 23: Real-world hand-arm manipulation platform. The system comprises a 16 DoF hand mounted on a 6 DoF Sawyer arm. The goal of the task is to perform in-hand reorientation, as illustrated in Fig 28 or pipe insertion as shown in Fig 29

We also evaluate MTRF on the real-world hand-arm robotic system, training a set of tasks in the real world, without any simulation or special instrumentation. We considered 2 different task families - in hand manipulation of a 3 pronged valve object, as well as pipe insertion of a cylindrical pipe into a hose attachment mounted on the wall. We describe each of these setups in detail below:

In-Hand Manipulation: For the first task on hardware, we use a variant of the in-hand reorienting task, where the goal is to pick up an object and reorient it in the palm into a desired configuration, as shown in Fig 24. This task not only requires mastering the contacts required for a successful pickup, but also fine-grained finger movements to reorient the object in the palm, while at the same time balancing it so as to avoid dropping. The task graph corresponding to this domain is shown in Fig 24. A frequent challenge in this domain stems from dropping the object during the in-hand reorientation, which ordinarily would require some sort of reset mechanism (as seen in prior work Nagabandi et al., 2020). However, MTRF enables the robot to utilize such "failures" as an opportunity to practice the tabletop re-centering, pickup, and flip-up tasks, which serve to "reset" the object into a pose where the reorientation can be attempted again.[1] The configuration of the 22-DoF hand-arm system mirrors that in simulation. The object is tracked using a motion capture system. Our policy directly controls each joint of the hand and the position of the end-effector. The system is set up to allow for extended uninterrupted operation, allowing for over 60 hours of training without any human intervention. We show how our proposed technique allows the robot to learn this task in the following section.

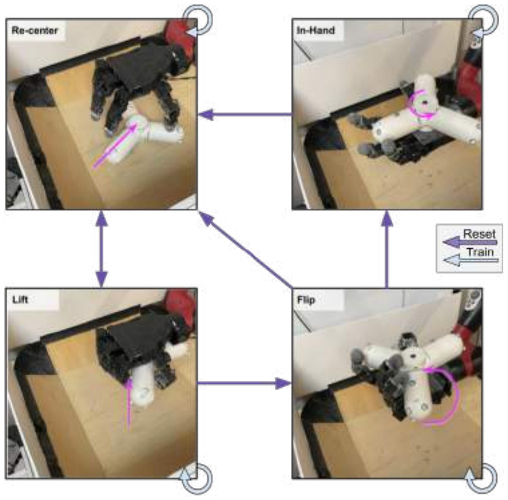


Figure 24: Tasks and transitions for the in-hand manipulation domain on hardware. The goal here is to rotate a 3 pronged valve object to a particular orientation in the palm of the hand, picking it up if it falls down to continue practicing.

Pipe insertion: For the second task on hardware, we set up a pipe insertion task, where the goal is to pick up a cylindrical pipe object and insert it into a hose attachment on the wall, as shown in Fig 25. This task not only requires mastering the contacts required

1 For the pickup task, the position of the arm's end-effector is scripted and only D'Hand controls are learned to reduce training time.

for a successful pickup, but also accurate and fine-grained arm motion to insert the pipe into the attachment in the wall. The task graph corresponding to this domain is shown in Fig 25. In this domain, the agent learns to pickup the object and then insert it into the attachment in the wall. If the object is dropped, it is then re-centered and picked up again to allow for another attempt at insertion. [2] As in the previous domain, our policy directly controls each joint of the hand and the position of the end-effector. The system is set up to allow for extended uninterrupted operation, allowing for over 30 hours of training without any human intervention.

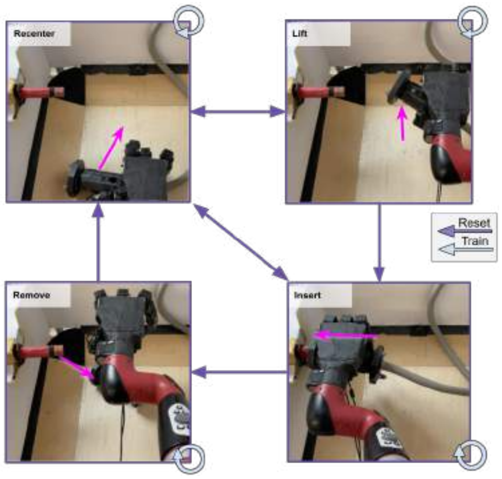


Figure 25: Tasks and transitions for pipe insertion domain on hardware. The goal here is to reposition a cylindrical pipe object, pick it up and then insert it into a hose attachment on the wall.

5.5 EXPERIMENTAL EVALUATION

We focus our experiment on the following questions:

1. Are existing off-policy RL algorithms effective when deployed under reset-free settings to solve dexterous manipulation problems?
2. Does simultaneously learning a collection of tasks under the proposed multi-task formulation with MTRF alleviate the need for resets when solving dexterous manipulation tasks?

2 For the pickup task, the position of the arm's end-effector is scripted and only D'Hand controls are learned to reduce training time. For the insertion task, the fingers are frozen since it is largely involving accurate motion of the arm.

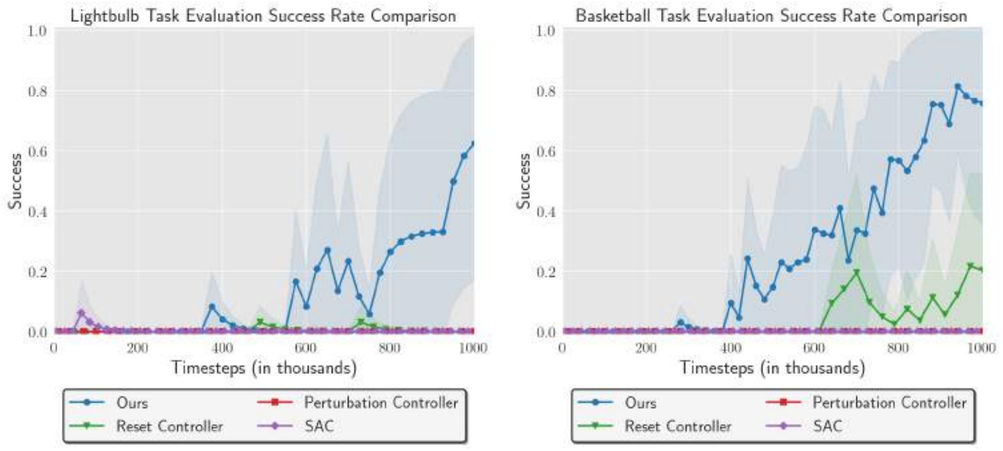


Figure 26: Comparison of MTRF with baseline methods in simulation when run without resets. In comparison to the prior reset-free RL methods, MTRF is able to learn the tasks more quickly and with higher average success rates, even in cases where none of the prior methods can master the full task set. MTRF is able to solve all of the tasks without requiring any explicit resets.

3. Does learning multiple tasks simultaneously allow for reset-free learning of more complex tasks than previous reset free algorithms?
4. Does MTRF enable real-world reinforcement learning without resets or human interventions?

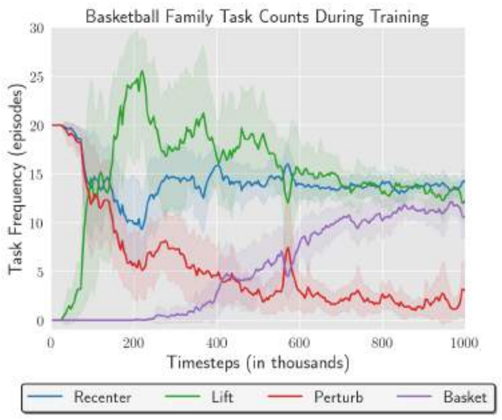


Figure 27: Visualization of task frequency in the basketball task Family. While initially recentering and pickup are common, as these get better they are able to provide resets for other tasks.

5.5.1 *Baselines and Prior Methods*

We compare MTRF (Section 6.4) to three prior baseline algorithms. Our first comparison is to a state-of-the-art off-policy RL algorithm, soft actor-critic Haarnoja et al., 2018c (labeled as *SAC*). The actor is executed continuously and reset-free in the environment, and the experienced data is stored in a replay pool. This algorithm is representative of efficient off-policy RL algorithms. We next compare to a version of a reset controller Eysenbach et al., 2017 (labeled as *Reset Controller*), which trains a forward controller to perform the task and a reset controller to reset the state back to the initial state. Lastly, we compare with the perturbation controller technique Zhu et al., 2020b introduced in prior work, which alternates between learning and executing a forward task-directed policy and a perturbation controller trained purely with novelty bonuses Burda et al., 2019 (labeled as *Perturbation Controller*). For all the experiments we used the same RL algorithm, soft actor-critic Haarnoja et al., 2018c, with default hyperparameters. To evaluate a task, we roll out its final policy starting from states randomly sampled from the distribution induced by all the tasks that can transition to the task under evaluation, and report performance in terms of their success in solving the task.

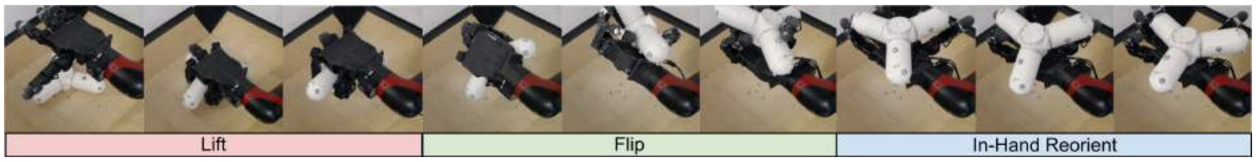


Figure 28: Film strip illustrating partial training trajectory of hardware system for in-hand manipulation of the valve object. This shows various behaviors encountered during the training - picking up the object, flipping it over in the hand and then in-hand manipulation to get it to a particular orientation. As seen here, MTRF is able to successfully learn how to perform in-hand manipulation without any human intervention.

5.5.2 *Reset-Free Learning Comparisons in Simulation*

We present results for reset-free learning, using our algorithm and prior methods, in Fig 26, corresponding to each of the tasks in simulation in Section 5.4. We see that MTRF is able to successfully learn all of the tasks jointly, as evidenced by Fig 26. We measure evaluation performance after training by loading saved policies and running the policy corresponding to the "forward" task for each of the task families (i.e. lightbulb insertion and basketball dunking). This indicates that we can solve all the tasks, and as a result

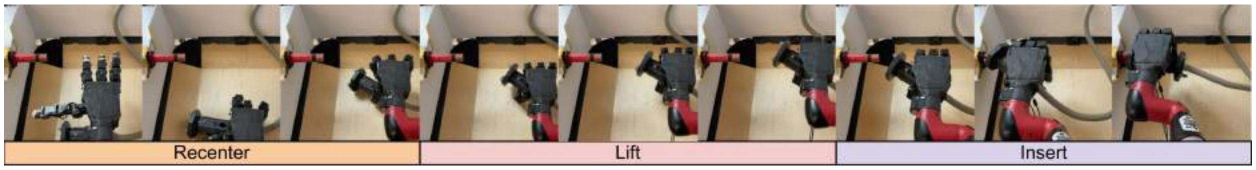


Figure 29: Film strip illustrating partial training trajectory of hardware system for pipe insertion. This shows various behaviors encountered during the training - repositioning the object, picking it up, and then inserting it into the wall attachment. As seen here, MTRF is able to successfully learn how to do pipe insertion without any human intervention.

can learn reset free more effectively than prior algorithms.

In comparison, we see that the prior algorithms for off-policy RL – the reset controller Eysenbach et al., 2017 and perturbation controller Zhu et al., 2020b – are not able to learn the more complex of the tasks as effectively as our method. While these methods are able to make some amount of progress on tasks that are constructed to be very easy such as the pincer task family shown in Appendix B, we see that they struggle to scale well to the more challenging tasks (Fig 26). Only MTRF is able to learn these tasks, while the other methods never actually reach the later tasks in the task graph.

To understand how the tasks are being sequenced during the learning process, we show task transitions experienced during training for the basketball task in Fig 27. We observe that early in training the transitions are mostly between the recenter and perturbation tasks. As MTRF improves, the transitions add in pickup and then basketball dunking, cycling between re-centering, pickup and basketball placement in the hoop.

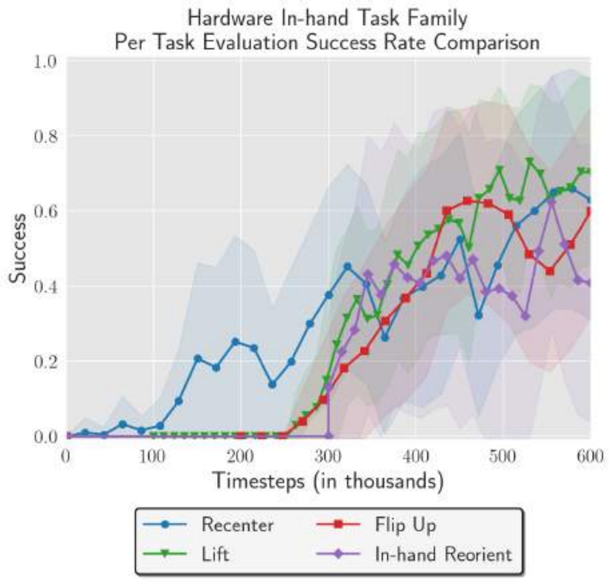
5.5.3 *Learning Real-World Dexterous Manipulation Skills*

Next, we evaluate the performance of MTRF on the real-world robotic system described in Section 5.4, studying the dexterous manipulation tasks described in Section 5.4.2.

IN-HAND MANIPULATION Let us start by considering the in-hand manipulation tasks shown in Fig 24. This task is challenging because it requires delicate handling of finger-object contacts during training, the object is easy to drop during the flip-up and in-hand manipulation, and the reorientation requires a coordinated finger gait to rotate the object. In fact, most prior work that aims to learn similar in-hand manipulation behaviors either utilizes simulation or employs a hand-designed reset procedure, potentially involving human interventions Nagabandi et al., 2020; Zhu et al., 2019; Kumar et al., 2016b. To the best of our knowledge, our work is the first to show a real-world robotic system learning such a task entirely in the real world and without any manually provided or hand-designed reset mechanism. We visualize a sequential execution of the tasks (after training) in Fig 28. Over the course of training, the robot must first learn to recenter the object on the table, then learn to pick it up (which requires learning an appropriate grasp and delicate control of the fingers to maintain grip), then learn to flip up the object so that it rests in the palm, and finally learn to perform the orientation. Dropping the object at any point in this process requires going back to the beginning of the sequence, and initially most of the training time is spent on re-centering, which provides resets for the pickup. The entire training process takes about 60 hours of real time, learning all of the tasks simultaneously. Although this time requirement is considerable, it is entirely autonomous, making this approach scalable even without any simulation or manual instrumentation. The user only needs to position the objects for training, and switch on the robot.

For a quantitative evaluation, we plot the success rate of sub-tasks including re-centering, lifting, flipping over, and in-hand reorientation. For lifting and flipping over, success is defined as lifting the object to a particular height above the table, and for reorient success is defined by the difference between the current object orientation and the target orientation of the object. As shown in Fig 30, MTRF is able to autonomously learn all tasks in the task graph in 60 hours, and achieves an 70% success rate for the in-hand reorient task. This experiment illustrates how MTRF can enable a complex real-world robotic system to learn an in-hand manipulation behavior while at the same time autonomously retrying the task during a lengthy unattended training run, without any simulation, special instrumentation, or manual interventions. This experiment suggests that, when MTRF is provided with an appropriate set of tasks, learning of complex manipulation skills can be carried out entirely autonomously in the real world, even for highly complex robotic manipulators such as multi-fingered hands.

PIPE INSERTION We also considered the second task variant which involves manipulating a cylindrical pipe to insert it into a hose attachment on the wall as shown in Fig 25. This task is challenging because it requires accurate grasping and repositioning of the object during training in order to accurately insert it into the hose attachment, requiring coordination of both the arm and the hand. Training this task without resets requires a combination of repositioning, lifting, insertion and removal to continually keep training and improving. We visualize a sequential execution of the tasks (after training) in Fig 29. Over the course of training, the robot must first learn to recenter the object on the table, then learn to pick it up (which requires learning an appropriate grasp), then learn to actually move the arm accurately to insert the pipe to the attachment, and finally learn to remove it to continue training and practicing. Initially most of the training time is spent on re-centering, which provides resets for the pickup, which then provides resets for the insertion and removal. The entire training process takes about 25 hours of real time, learning all of the tasks simultaneously.



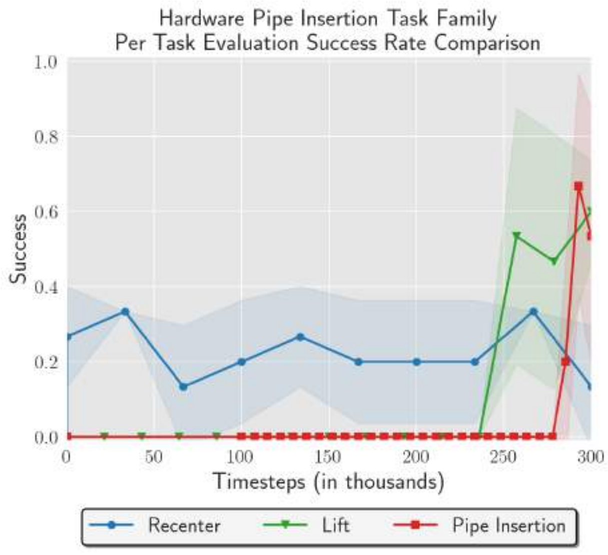


Figure 30: Success rate of various tasks on dexterous manipulation task families on hardware. **Left: In-hand manipulation** We can see that all of the tasks are able to successfully learn with more than 70% success rate. **Right: Pipe insertion** We can see that the final pipe insertion task is able to successfully learn with more than 60% success rate.

5.6 DISCUSSION

In this work, we introduced a technique for learning dexterous manipulation behaviors reset-free, without the need for any human intervention during training. This was done by leveraging the multi-task RL setting for reset-free learning. When learning multiple tasks simultaneously, different tasks serve to reset each other and assist in uninterrupted learning. This algorithm allows a dexterous manipulation system to learn manipulation behaviors uninterrupted, and also learn behavior that allows it to continue practicing. Our experiments show that this approach can enable a real-world hand-arm robotic system to learn an in-hand reorientation task, including pickup and repositioning, in about 60 hours of training, as well as a pipe insertion task in around 25 hours of training without human intervention or special instrumentation.

6

LEARNING MULTI-TASK RESET-FREE BEHAVIOR FROM IMAGES

As seen in Chapter 5, running RL on real-world robotic platforms raises a number of issues that lie outside the standard RL formulation, including but not limited to the need for episodic resets. Other challenges include the problem of state estimation in open world environments, and difficulties with rewards specification. In order to allow for learning-based dexterous manipulation systems to reach their fullest potential in terms of practicality and scalability, it is critical to limit the assumptions on manual engineering while still providing enough supervision for reinforcement learning to be tractable.

In this chapter, we ask: how we can enable reinforcement learning under assumptions that might plausibly hold outside of laboratory settings? And furthermore, how we can do so in such a way that tasks can be defined without manually programming well-shaped reward functions? Outside of laboratory settings, RL-enabled robots must deal not only with sample complexity, but also with the inability to reset the environment, and the need to deduce both object state and rewards from images obtained from on-board cameras. While many of these challenges can be tackled by leveraging human supervision, such as demonstrations, explicit interventions, or manually engineered shaped rewards, a practical real-world reinforcement learning system requires striking the right balance between obtaining human supervision that makes the autonomous RL problem tractable and not placing an untenable burden of supervision on the user. For instance, while shaped reward functions make the RL problem far more tractable, they impose a significant engineering burden.

Guided by this principle that costly human guidance should serve to minimally illustrate *how* to enable intervention-free learning, we propose a system that continuously collects experience guided by user-provided intermediate "milestone" examples that both provide guidance in how to perform the task, and specify how it can be practiced with autonomous retrying. To illustrate, consider a robotic hand whose goal is to grasp a hose connector and plug it into a socket (Fig 35). Training this task requires repeatedly plugging in and unplugging the hose connector, and the task definition should additionally provide enough information about the steps involved in performing the task so that the robot can learn it efficiently. In this case, the user might specify a milestone where the hand is holding the connector by positioning the hand over the object, another milestone where the plug is lined up with the socket, and a final one where it is fully inserted. Note, however, that the user does not need to teleoperate the robot to do the task – the individual steps can simply be "posed" by the user to define the milestones, and the system can then learn how to actually perform them. In this sense, the milestones are closer to step-by-step instructions, rather than full demonstrations.

The principle contribution of this paper is a system for real-world, high-dimensional, dexterous manipulation that avoids the need for teleoperation, manually engineered reset mechanisms, or environment information beyond the robot's own onboard sensors and vision. We show that, by instead leveraging user supervision in the form of visual milestones, which can be generated by a user taking a snapshot of the robot and object in a desired setting, our system can automate the procedure of both reset-free practicing and reward specification. We demonstrate our approach on a group of real world tasks, where, through over 80 hours of combined (but fully autonomous) robot interaction, we show that it is possible to learn a collection of contact rich tasks, paving the way for increasingly open-ended, real world, dexterous manipulation systems.

6.1 RELATED WORK

The design and control of anthropomorphic robot hands has been studied extensively in the robotics literature, varying in the complexity of the robot morphology and the degree of environment instrumentation (Xu et al., 2013; Deimel and Brock, 2016; Gupta et al., 2016b). Prior work has studied control of complex hands using trajectory optimization (Mordatch et al., 2012; Kumar et al., 2014b), policy search (Kober and Peters, 2008; Posa et al., 2014; Rajeswaran et al., 2017b), simulation to real-world transfer OpenAI et al., 2018; Lowrey et al., 2018; Allshire et al., 2021, and reinforcement learning (Van Hoof et al., 2015; Zhu et al., 2019). In contrast to our work, the majority of this prior work has assumed access to compact state representations or accurate simulators and object models. Closer to the system we describe in this paper is prior work on learning visuomotor policies for dexterous manipulation (Jain et al., 2019; Mandikal and Grauman, 2020b; Akinola et al., 2020), which shares our motivation for reducing the assumptions required for real world learning. However, with the exception of several works we discuss below, prior systems on RL for dexterous manipulation typically require assumptions on user-provided resets, manually designed rewards, or ground truth object state observations. These assumptions hinder the application of RL in more realistic settings.

An important consideration in our system is the ability to specify an task without manual reward engineering via intermediate milestone examples. Task specification has been studied extensively, but, however, using a variety of assumptions and approaches. These range from having humans provide demonstrations for enabling imitation learning Argall et al., 2009; Ross et al., 2013; Reddy et al., 2019, using inverse RL Ziebart et al., 2008a; Wulfmeier et al., 2015b; Ratliff et al., 2006b, active settings where users can provide corrections Losey and O'Malley, 2018; Cui and Niekum, 2018; Co-Reyes et al., 2018, or ranking-based preferences Myers et al., 2022; Brown et al., 2020. Motivated by the goal of broader applicability, we do not assume access to expert demonstrations (e.g. via teleoperation or kinesthetic teaching), which can themselves can be difficult to provide for high-dimensional systems Akgun et al., 2012; Villani et al., 2018. For instance, providing kinesthetic demonstrations for a full hand-arm robotic system requires very challenging coordination and several simultaneous demonstrators. Instead, we utilize sparse images of intermediate outcomes that can be obtained simply by posing the robot and objects in a particular state, which is generally much easier to provide than kinesthetic demonstrations. For reward inference using these intermediate outcome states, we build on the VICE framework Fu et al., 2018b, although our system could use any reward inference algorithm based upon success examples Zolna et al., 2020. However, our focus is not on devising a new *algorithm* for learning rewards, but on leveraging existing components, such as VICE, to build a complete, reset-free, robotic system that can enable scalable RL with a dexterous manipulator in the real world. Therefore, although the building blocks of our system are based on prior work, their combination and the capabilities they enable (learning image-based dexterous manipulation in the real world) are novel.

We build on prior work that has studied the setting of "reset-free" reinforcement learning (see Sharma et al. (2021b) for a review). Prior work has studied this setting from a number of complementary perspectives, such as safety (Eysenbach et al., 2017), skill discovery Xu et al., 2020, unattended learning (Han et al., 2015) and via curriculum (Sharma et al., 2021a). Like these prior works, our system is also aimed at enabling "reset-free" learning. The majority of this prior work, however, relies on compact state representations, which require manual engineering to construct in the real world (using systems like motion capture or object trackers). In contrast, our system overall has substantially less restrictive assumptions (raw vision-based observations, learning via intermediate outcome states, multi-task learning).

The most closely related reset-free robotic RL systems that have been previously proposed are R3L (Zhu et al., 2020b) and MTRF (Gupta et al., 2021b). Our assumptions regarding minimal instrumentation and vision-based reset-free learning most resemble those of Zhu et al. (2020b) (R3L). However, our work tackles a considerably more challenging setting: while Zhu et al. (2020b) studied a 3-finger claw mounted on a fixed base, we show that our method can control a 4-fingered hand on a 7 DoF arm. This is done by leveraging a multi-task RL formulation that builds on ideas from MTRF (Gupta et al., 2021b) instead of the novelty-based resets in R3L (Burda et al., 2019), which scale poorly in higher dimensional settings. In contrast to MTRF, however, which requires known low-dimensional state and hand-designed shaped rewards, our system learns vision-based rewards from examples, and does not require motion capture or user-provided state estimation systems. Thus, our system can be seen as combining the strengths of R3L (Zhu et al., 2020b) and MTRF (Gupta et al., 2021b), and addressing the major limitations of both systems.

Table 3: A comparison between the assumptions of AVAIL and prior reset-free methods.

Method	**No Dense Reward**	**Multi Task**	**Vision**	**High DoFs**
R3L	✓	×	✓	×
MTRF	×	✓	×	✓
Ours	✓	✓	✓	✓

6.2 ROBOTIC PLATFORM AND PROBLEM OVERVIEW

To contextualize our problem setting, we first present an overview of our robot platform, describing the hardware and task setup, as well as the observation and action space. Then, we provide an overview of our problem setting, focusing on the practical goals of our system. We provide details related to our robotic platform in the supplementary material under Appendix E, along with details of a simulated analogue that we employ for analysis and ablation experiments.

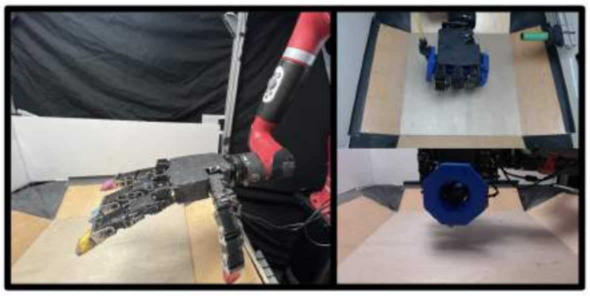


Figure 31: The robot consists of a 16 DoF hand mounted on a 7 DoF Sawyer arm (left). The workspace (right) is equipped with two RGB web cameras providing an overhead and side view.

We study our approach in the challenging domain of reset-free dexterous manipulation, where we make use of a custom, anthropomorphic manipulation platform. Our system consists of a custom-built, 4 finger, 16-DoF robot hand, mounted on a 7-DoF Sawyer robotic arm. The arm and hand assembly is positioned over a tabletop surface (Fig. 31, left image). Our policy, which we operate at 8Hz, directly controls each joint position in addition to the Cartesian position and orientation of the arm, resulting in a 22 dimensional action space and 29 dimensional state space. The system is designed to operate for upwards of 100 hours in contact-rich environments without breakage. In addition to the robot's own joint encoders, two RGB image observations are provided to the robot via two low-cost web cameras (Fig. 31, right image, red boxes) positioned in third-person positions and resized to 84×84.

Our tasks consist of manipulation behaviors such as reaching, grasping, reorienting and inserting. We consider two different collections of tasks for interacting with several different objects: inserting a hose into a connector on the side of the arena, and hooking a rope onto a fixture. For each of these tasks, the only supervision interface we assume is to allow the user to place the robot and object in a desired position and capture a set of image "snapshos". We describe how we use this supervision to drive reward inference and task selection in the sections below.

6.3 PROBLEM FORMALISM AND USER ASSUMPTIONS

In this section, we formalize our reset-free problem setting and supervision assumptions. Consider first the Markov decision process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, p_{dyn}, \rho, \gamma, R)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action spaces, $p_{dyn} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}_{\geqslant 0}$ denotes the environment dynamics, $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ denotes the reward function, $\rho : \mathcal{S} \mapsto \mathbb{R}_{\geqslant 0}$ denotes the initial state distribution and $\gamma \in [0, 1)$ denotes the discount factor. The typical objective of episodic RL is to optimize the discounted return $J(\pi) = \mathbb{E}_{\tau \sim \pi}[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)]$ with respect to the policy π, where $\tau = \{(s_i, a_i)\}_{i=0}^{T-1}$ is obtained by sampling $s_0 \sim \rho(\cdot)$, $a_t \sim \pi(\cdot \mid s_t)$ and $s_{t+1} \sim p(\cdot \mid s_t, a_t)$.

Standard episodic RL assumes the ability to sample from and *set* the environment to $s_0 \sim \rho$ between trials. While trivial to satisfy in simulated settings, this need to effectively teleport the environment state to s_0 between attempts has substantial practical cost in the real world. As typical RL algorithms require episodic resets in order to learn effectively Sharma et al., 2021a, this often requires expensive human engineering via a separately constructed reset mechanism Zhu et al., 2019; Nagabandi et al., 2020. This need for manual engineering can in principle be avoided by learning a separate policy whose goal is to "reset" the environment Eysenbach et al., 2017; Han et al., 2015. However, in many realistic settings, performing a "reset" can be as challenging as the original task, and can entail a range of different behaviors. For example, a robot that uses a coffee machine might spill the coffee, requiring it to clean up the spill. It might also drop the cup, requiring it to reach for it. Not only are each of these tasks themselves difficult to solve with a single policy, they are also sufficiently distinct from each other that we might want entirely different strategies. In addition, independent of the need to be sequenced in a reset-free setting, these tasks are themselves challenging to learn which necessitates the provision of more fine-grained guidance via user supervision.

To be able to address these challenges in reset-free task sequencing and providing fine-grained supervision, we propose a more nuanced strategy, where a user provides the robot with a set of sub-problems to practice. We propose to provide the robot with human supervision using example "milestones" summarized in the following graph structure:

Definition 1 (*Milestones graph*). *We assume the user provides a set of outcome images that can be summarized by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of cardinality $|K|$ indexed by z, where each vertex $v \in \mathcal{V}$ is characterized by a collection of outcome images s_g^z (which indicate a semantically meaningful subtask to be solved). Each set of outcome images characterizes a separate "sub-task". In addition, upon successfully accomplishing a sub-task, a directed edge $(v, v') \in \mathcal{E}$ is provided which indicates which sub-task is to be practiced next.*

Consistent with the goal of having the agent continuously practice (i.e., not get stuck), we assume there are no sink nodes in the provided graph[1]. Then, instead of optimizing a single-task objective $J(\pi)$, we instead optimize all of the subtasks in the milestone graph simultaneously, resulting in a multi-task RL problem. Concretely, we now learn a set of K policies π_z indexed by a categorical variable z (one for each milestone), optimizing a set of MDPs, $\mathcal{M} \equiv (\mathcal{S}, \mathcal{A}, p_{dyn}(s_{t+1}|a_t, s_t), \{R_z\}_{z=0}^{K-1}, p_{task}(z|s))$, where we have introduced a per milestone reward R_z and task predictor p_{task}, using which we can construct the following objective

$$J_{MT}(\{\pi_i\}_{i=0}^{K-1}) = \sum_{i=0}^{\infty} \left[\mathop{\mathbb{E}}_{\substack{s_0^i = s_T^{i-1} \\ \tau \sim \pi_{z_i}}} \left[\sum_{t=0}^{T} \gamma^t R_z(s_t^i, a_t^i) \right] \right] \tag{17}$$

$$z_{i+1} \sim p_{task}(z_{i+1}|s_T^i). \tag{18}$$

Formulating the problem in this manner makes conspicuous the need to define and sample from $p_{task}(z_{i+1}|s_T^i)$ and $R_z(s_t, a_t)$. To tractably learn these two functions, we leverage the user provided milestone supervision, which we denote as $D_k = \{s_1, s_2, ..s_N\}$, and a

1 This is a strong assumption, and conceivably could be lifted by providing handling of safety-critical states to avoid irreversible sinks, (Garcıa and Fernández, 2015; Srinivasan et al., 2020). For simplicity we leave addressing safety issues for future work.

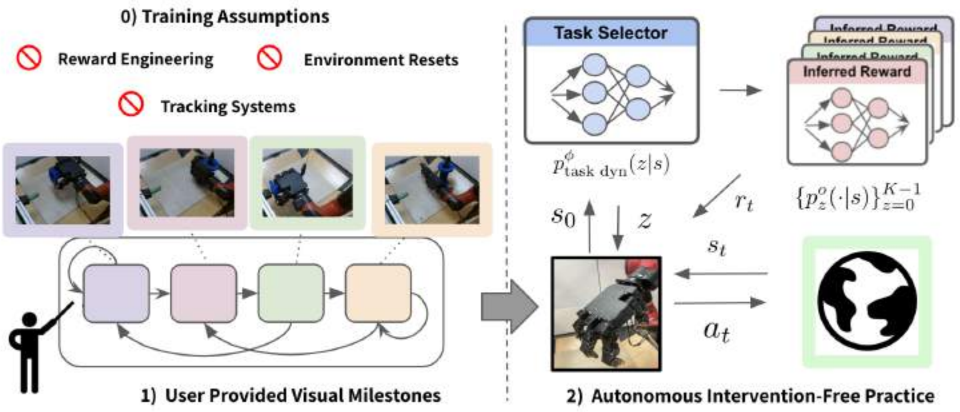

0) Training Assumptions
Reward Engineering
Environment Resets
Tracking Systems
1) User Provided Visual Milestones
Task Selector
$p^{\phi}_{\text{task dyn}}(z|s)$
Inferred Reward
$\{p^{o}_{z}(\cdot|s)\}_{z=0}^{K-1}$
s_0
z
r_t
s_t
a_t
2) Autonomous Intervention-Free Practice

Figure 32: An overview of our approach. Our relies on the human to provide a set of visual milestone (bottom left, $K = 4$ here) and transitions. We use this to formulate a multi-task learning problem where we leverage this supervision to learn all the components necessary for intervention-free learning. Prior to training, we learn a task selection model (blue module), which is used to choose amongst a set of K policies for data collection (Sec. 6.4.2). This data is used to learn a success classifier (top right, Sec. 6.4.1), which is used to automatically assign rewards. We show that our system is capable of learning reset-free, complex manipulation behaviors on a real world anthropomorphic hand without instrumentation beyond the robot's own joint encoders and vision.

set of categorical labels y_k for each set of success examples indicating which task should transition to the next. These could, for example, be a set of images showing the robot repositioning and re-grasping the object, and a label indicating the next step is pickup. In the following sections, we show how to specify the rewards R_z and learn the task transition function $p_{\text{task}}(z_{i+1}|s^i_T)$ and per-task policies π_z without any human intervention in the training process, simply using milestone image supervision provided upfront.

6.4 THE AVAIL SYSTEM: AUTONOMY VIA USER-PROVIDED MILESTONES

To address the problem described in Section 6.3, we present AVAIL (Autonomy ViA mILestones) – our system for learning autonomously with minimal intervention and minimal external environment instrumentation. AVAIL solves a reset-free autonomous RL problem by reframing the RL training process as a multi-task, reset-free problem that can learn directly from raw visual inputs, with minimal external instrumentation or intervention. By leveraging the user-provided milestones defined in Sec 6.3, at a high level our system functions by (1) deriving reward functions via learned success classifiers, (2) optimizing these rewards in a sample efficient manner using a multi-task vision-based RL system, and (3) determining which of these tasks to perform given the current robot observations.

6.4.1 *Visual Multi-Task Policy and Reward Learning from User Milestones*

A critical enabler of autonomous learning is the ability for a robot to assign rewards to its own experience. This importantly relieves the burden of manual reward engineering, but comes with the trade-off of removing the ability of the human to provide task information (e.g., via reward shaping), which can be critical for tractable learning of long horizon tasks. To resolve this difficult challenge in our setting, which effectively requires compound behavior over long horizons to attempt and reset the task, we leverage the sparse milestone supervision to learn a set of success classifiers $\{p_z^o(\cdot|s)\}_{z=0}^{K-1}$. Importantly, the provision of a number of significant milestones takes the burden off a single learned classifier to provide accurate reward shaping for long horizon tasks. By breaking down a long horizon behavior into a number of component milestones, our system is able to deal with relatively poor reward shaping from the learned classifier for learning individual policies.

For each individual classifier $p_z^o(\cdot|s)$, we make use of the VICE algorithm Fu et al., 2018b. Concretely, we learn a binary classifier $p_z^o(\cdot|s)$ over the set of user-provided examples $\{D_1, D_2, \dots, D_K\}$ as the positive class, and the agent's own experience sampled on-policy as the negative class. Once trained, the classifier probability $p_z^o(o|s)$, or a monotonic transformation of it (e.g., $\log p_z^o(\cdot|s)$), can be used as R_z from Section 6.3. Combined with our manner of breaking down the problem into concrete milestones, an added advantage of classifier-based rewards is the property that, in practice, they can often provide some additional degree of shaping near the success Fu et al., 2018b; Li et al., 2021.

Finally, in order to learn in uninstrumented settings, a core component of our system is a sample efficient, multi-task RL component that allows us to learn the robot's joint encoders and raw image observations. We build on recent advances in data augmentation Kostrikov et al., 2020 and implicit ensembling Hiraoka et al., 2021 via dropout to allow for robust, sample efficient learning. In particular, given the set of reward functions discussed above, we learn a set of corresponding K policies π_z using the recently proposed DRoQ approach (Hiraoka et al., 2021) which combines dropout and data-augmentation for vision based RL.

6.4.2 *Multi-Task Learning without Oracles*

After attempting a particular task, the agent must decide which task to attempt next, which depends on the current situation (e.g., if it drops the hose connector, it should try to grasp it again, but if it is still holding it, it can attempt the insertion). In order to infer which task the agent should execute, we allow the user to provide next milestone labels (labels of what discrete milestone z_k should be attempted at a particular state s_k) to train a parameterized task dynamics model by performing supervised learning over the milestones and labels provided at the beginning of training. That is, given a dataset of $\mathcal{D} = \{(z_k, s_k)_{k=0}^{N}\}$, we can recover $p_{\text{task}}^{\phi} = \arg\max_{\phi} \mathbb{E}_{z_k, s_k \sim \mathbf{D}} \left[\log p_{\text{task}}^{\phi}(z_k|s_k)\right]$. These labels represent which task to perform upon success (e.g., the success examples for hooking the rope could be labeled with the 'unhook' task). During autonomous training, the agent samples a task from this learned model, which is then used to execute the corresponding policy π_z in the environment. Rather than re-sample this task indicator at every step, we sample it every T steps and keep it fixed during data collection. This scheme explicitly separates the task inference, and task learning allows each "sub-problem" to be treated effectively as a separate MDP.

6.4.3 *Algorithm Summary and Implementation Details*

To summarize, given the milestone graph provided by the users, our system, AVAIL (Autonomy ViA mILestones), proceeds as follows. First, AVAIL performs supervised learning of the next task transitions provided by the user as described in Sec. 6.4.2 to learn $p_{\text{task}}^{\phi}(z|s)$. Next, during our training, our approach chooses the most probable task z using an observed state, which is then used to collect experience using the corresponding policy π_z. We train a set of separate policies π_z for each of the K sets of example images, with separate critics Q_z and replay buffers $\mathcal{B}_z$. We parameterize each policy π_z as a deep neural network, and train each policy using the soft actor-critic algorithm (SAC) Haarnoja et al., 2018c using rewards R_z inferred via VICE Fu et al., 2018b. Finally, rather than resampling the task every step, we do so every $T = 100$ steps. We summarize our full approach in Alg. E.3.1 and provide more extensive details and hyperparameters in Appendix E.3.

6.5 EXPERIMENTAL EVALUATION

Our experiments first aim to evaluate whether AVAIL can autonomously learn complex manipulation skills in the real world with visually indicated milestones, and then to compare AVAIL with other reset-free learning methods in simulation.

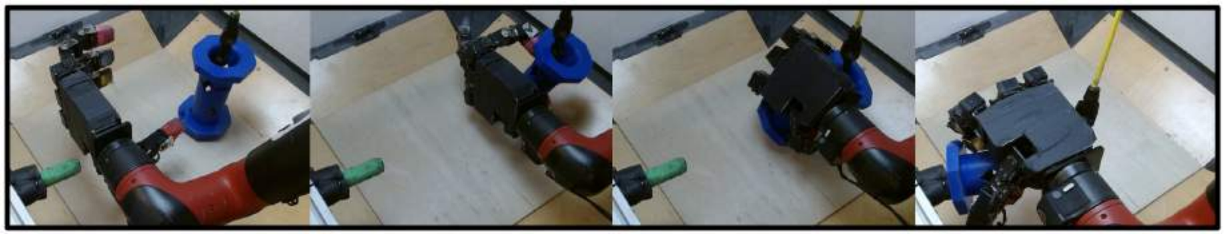


Figure 33: Filmstrip of final learned insertion behavior. Using the user-provided milestones, our robot learns a set of skills (e.g., grasp, insert), which together specify a long horizon task as well as the stages needed to practice autonomously. In our experiment, the robot learns to successfully plug-in a hose connector over 36 hours of continuous learning without instrumentation, purely from image observations and joint encoders.

Real-world tasks and evaluation: Since the principle aim of AVAIL is to acquire real world skills, we evaluate our approach on two manipulation tasks in settings without any environment instrumentation, handcrafted reward functions, or engineered reset mechanisms. The first task involves grasping, hooking, and unhooking a rope onto a handle, and the second involves grasping, repositioning, and connecting a cylindrical hose connector to a plug in the workspace. We periodically save the policies at regular intervals and evaluate their performance after training to not interrupt reset-free training. For both task sets, the evaluation metric for each milestone is a binary indicator of success based on the distance of the hand and object to the desired goal position. We provide more extensive details in the supplementary material under Appendix E.1.2 and E.1.3.

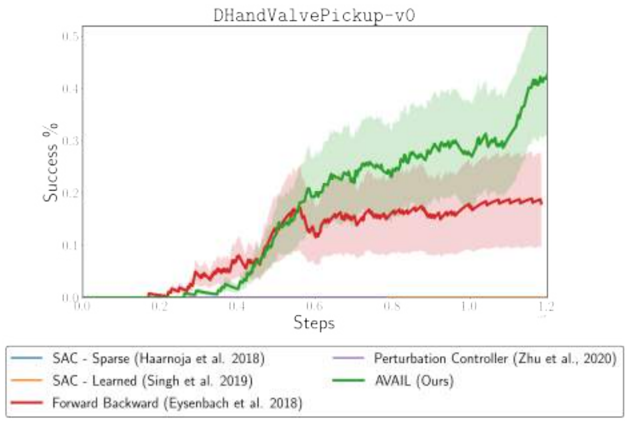
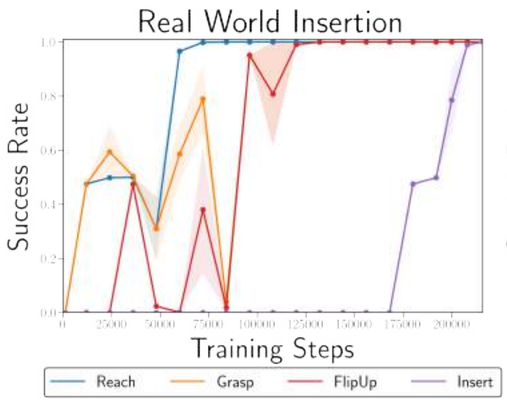
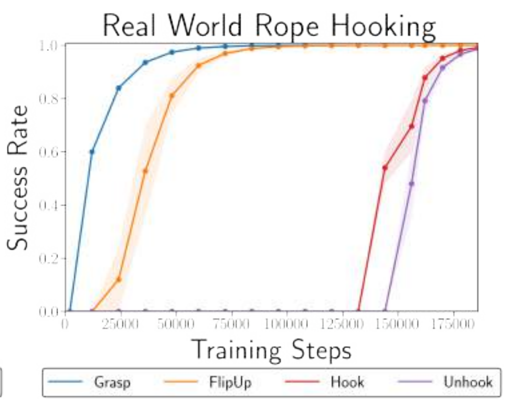


Figure 36: Success rates of each method averaged over 5 seeds for the full task on the `DHandValvePickup-v0` domain. Novelty based resets (purple) fail to make progress in this high DoF control problem. Compared to methods with fewer degree of supervision, $K = 0, 1, 2$ milestones (blue, orange, red), our results illustrate the benefits of milestone supervision.

We plot the performance of the evaluation runs as a function of the time at which the policy was recorded in Fig. 52, providing learning curves for real-world training. Observe that AVAIL automatically provides a degree of scaffolding by successfully learning skills early on in training (blue and orange) that correspond to being able to regrasp and reorient the object. This allows the robot to continuously retry inserting and hooking task. By the end of training, we find that the robot is able to successfully perform both tasks. We note that for these two tasks, no additional instrumentation is required beyond



(a)



(b)

Figure 34: Success rates of different milestones on our real world dexterous manipulation system. Our method is able to insert the pick with over 80% success rate at the end of training (right, red curve) and nearly perfectly hook and unhook the rope (left, red/green curve). Overall, success on other milestones improves earlier in training (blue, orange curves), equipping the robot with skills to autonomously retry the task.

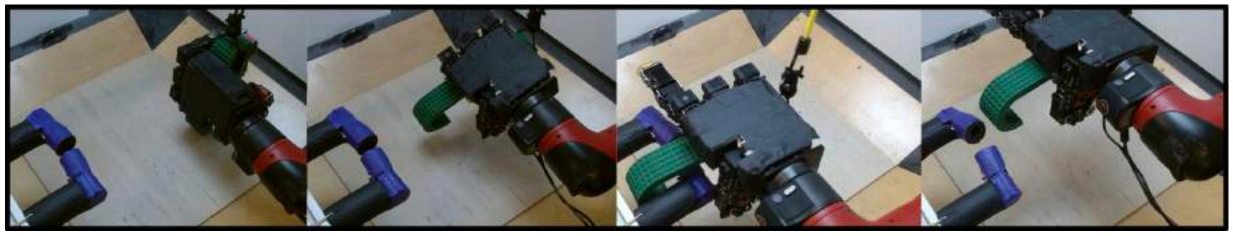


Figure 35: Filmstrip of the final learned rope hooking behavior. Using the user-provided milestones, our robot learns a set of skills that allows it to autonomously practice hooking and unhooking the rope (right two images) and recover from failure (e.g., after dropping the rope) by regrasping and reorienting the hook (left two images). After over 30 hours of unattended learning, our system successfully hooks the rope with over 95% success.

changing the object and fixture. Upon specifying the visual milestones, the robot is capable of completely unattended learning for approximately 80 hours of combined robot time.

Comparisons. In order to compare AVAIL more extensively to prior reset-free methods, we build on the D'Hand simulation domain developed in prior work Gupta et al., 2021b (see Appendix E.2.1). We first compare to a standard state-of-the-art RL algorithm, soft actor-critic Haarnoja et al., 2018c (which we denote as *SAC*), which we run reset-free in the environment using a reward learned by pooling example images of the final task. Note that prior works in robotics have used this type of approach, which corresponds to a combination of SAC with VICE, to address real-world robotics tasks (Singh et al., 2019), but this approach does not by itself address the reset-free challenges. Next, we compare to a version of the reset controller, which can be seen as providing two milestones: one to pick up the object, and one to place it back on the table. Finally, we compare to the perturbation controller Zhu et al., 2020b, where we provide the "forward" policy with a set of pickup goals and follow Zhu et al. (2020b) by interleaving training the forward policy with a "perturbation controller" trained with an intrinsic reward based on random network distillation bonus Burda et al., 2019. Details for all of the methods are in included in the supplementary materials under Appendix E.3.

Evaluation protocol. In order to evaluate the final performance of each method, we sample a random initial position of the object in the workspace and run the learned policy. We evaluate task success using a sparse indicator function that evaluates whether the object's center of mass is within a 0.1m radius of the target position above the table. We emphasize that these rollouts are not used for training, and are only used to measure performance. For all methods, we report the final success rate of each policy in picking up the valve. For additional details related to our environment setup and evaluation metrics, we refer the reader to the supplementary material in Appendix E.2.

Comparative analysis. The results of our comparative evaluation are presented in Fig. 54. Prior methods generally do not make successful progress on this task, due to the combination of reset-free training and the lack of a shaped reward. Without any handling of the reset-free setting, both variants of SAC fail to progress. While R3L in principle can handle the reset-free setting by perturbing the state between trials, the large, high-dimensional task simply provides too many ways for the purely novelty-seeking controller to modify the environment with a meaningful reset. The forward backward controller (red), which can be seen as an instantiation of our approach with two milestones, is the only prior method that succeeds in making progress. This suggests that the improved performance can be achieved through granular milestones, which is also suggested by the further degradation of performance when using a sparse (light blue), where the user provides no milestones.

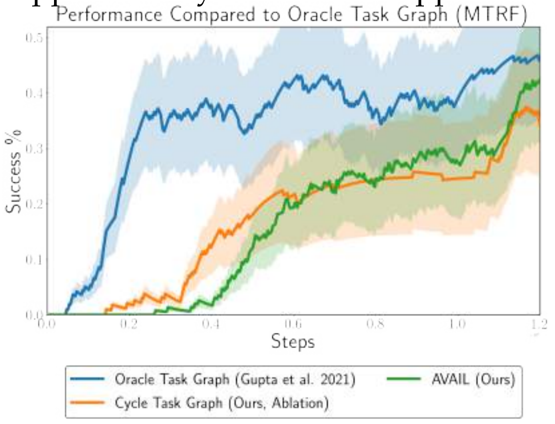


Figure 37: A comparison of success rate on the simulated `DHandValvePickup-v0` domain compared to an oracle task graph. We find that our framework is robust to "errors" in the task graph compared to a hand crafted oracle. Both a learned task graph and ablation perform similarly given enough training.

Task graph. Finally, to understand the effect of our learned task graph. We run an experiment in simulation where we can construct an "oracle" task graph similar to Gupta et al. (2021b), using ground truth state information about the object's position in the scene. We also compare with a naïve strategy for scheduling where each task of the three milestones are practiced in consecutive order. The comparative success rate can be seen in Fig. 37. Interestingly, we find that while learning with the ground truth task graph results in faster learning, both the learned and naïve task graph are able to achieve comparable performance, with the learned approach slightly outperforming the naïve approach. This suggests that our framework is robust to "errors" in the task graph, although learning can be made more optimal through improved scheduling of tasks.

6.6 DISCUSSION, LIMITATIONS, AND FUTURE WORK

We proposed a method for multi-task learning without resets directly from high dimensional image observations. Our method, AVAIL, constructs a task graph from a modest number of user-provided milestone examples. This task graph illustrates how to practice and reset the task, and provides guidance to the learning process in lieu of more standard manual reward shaping. Our method does not require manual state estimation, manual resets, or reward engineering, and while the milestone examples require human effort to provide, we expect in many cases that this effort is significantly lower than providing full demonstrations. Much like a teacher or coach might instruct a student not just by telling them the *goal* of a task but how they should go about practicing it, the milestone examples serve to provide guidance to the agent for how it should go about learning the desired behavior. Our experiments show that this approach effectively produces a setup where the agent first practices the easier tasks, and then builds up the more complex tasks on top of them, all the while learning autonomously without resetting.

Limitations: While our method provides a more scalable alternative to assuming full demonstrations or hand-designed rewards, it does have several limitations. We assume that both our task graph and reward functions can be specified with images, but this is not always practical in partially observed settings. Potential future work could consider other natural modalities for specifying milestones, such as language. Our method also learns separate policies for each milestone, and sharing representations across tasks could likely accelerate learning. Finally, the particular choice of milestones in practice require some user expertise in determining stages that are appropriate for the robot to learn. An exciting direction for future work could be to enable the robot to autonomously determine if a particular stage is too hard and ask for additional guidance or assistance.

7 BENCHMARKING ALGORITHMS FOR AUTONOMOUS REINFORCEMENT LEARNING

As we have seen, autonomous learning, especially for data hungry approaches such as reinforcement learning, is an important enabler of broader applicability. Rarely however, is it an explicit consideration, despite its clear desirability. Consider an example of a robot learning to clean and organize a home. We would ideally like the robot to be able to autonomously explore the house, understand cleaning implements, identify good strategies on its own throughout this process, and adapt when the home changes. This vision of robotic learning in real world, continual, non-episodic manner is in stark contrast to typical experimentation in reinforcement learning in the literature (Levine et al., 2016a; Chebotar et al., 2017; Yahya et al., 2017; Ghadirzadeh et al., 2017b; Zeng et al., 2020), where agents must be consistently reset to a set of initial conditions through human effort or engineering so that they may try again. In this final chapter, we propose to bridge the gap between the practical challenges of providing oracle resets and the normal assumptions of standard episodic RL. In particular, we provide a formalism and set of benchmark tasks that explicitly consider the challenges faced by agents situated in non-episodic environments, rather than treating them as being abstracted away by an oracle reset.

Our specific aim is to place a greater focus on developing algorithms under assumptions that more closely resemble autonomous real world robotic deployment. While the episodic setting naturally captures the notion of completing a "task", it hides the costs of assuming oracle resets, which when removed can cause algorithms developed in the episodic setting to perform poorly (Sec. 7.5.1). Moreover, while prior work has examined settings such as RL without resets (Eysenbach et al., 2017; Xu et al., 2020; Zhu et al., 2020a; Gupta et al., 2021b), ecological RL (Co-Reyes et al., 2020), or RL amidst non-stationarity (Xie et al., 2020) in isolated scenarios, these settings are not well-represented in existing benchmarks. As a result, there is not a consistent formal framework for evaluating autonomy in reinforcement learning and there is limited work in this direction compared to the vast literature on reinforcement learning. By establishing this framework and benchmark, we aim to solidify the importance of algorithms that operate with greater autonomy in RL.

Before we can formulate a set of benchmarks for this type of autonomous reinforcement learning, we first formally define the problem we are solving. As we will discuss in Section 7.3, we can formulate two distinct problem settings where autonomous learning presents a major challenge. The first is a setting where the agent first trains in a non-episodic environment, and is then "deployed" into an episodic test environment. In this setting, which is most commonly studied in prior works on "reset-free" learning (Han et al., 2015; Zhu et al., 2020a; Sharma et al., 2021a), the goal is to learn the best possible *episodic* policy after a period of non-episodic training. For instance, in the case of a home cleaning robot, this would correspond to evaluating its ability to clean a messy home. The second setting is a continued learning setting: like the first setting, the goal is to learn in a non-episodic environment, but there is no distinct "deployment" phase, and instead the agent must minimize regret over the duration of the training process. In the previous setting of the home cleaning robot, this would evaluate the persistent cleanliness of the home. We discuss in our autonomous RL problem definition how these settings present a number of unique challenges, such as challenging exploration.

The main contributions of our work consist of a benchmark for autonomous RL (ARL), as well as formal definitions of two distinct ARL settings. Our benchmarks combine components from previously proposed environments (Coumans and Bai, 2016; Gupta et al., 2019; Yu et al., 2020b; Gupta et al., 2021b; Sharma et al., 2021a), but reformulate the learning tasks to reflect ARL constraints, such as the absence of explicitly available resets. Our formalization of ARL relates it to the standard RL problem statement, provides a concrete and general definition, and provides a number of instantiations that describe how common ingredients of ARL, such as irreversible states, interventions, and other components, fit into the general framework. We additionally evaluate a range of previously proposed algorithms on our benchmark, focusing on methods that explicitly tackle reset-free learning and other related scenarios. We find that both standard RL methods and methods designed for reset-free learning struggle to solve the problems in the benchmark and often get stuck in parts of the state space, underscoring the need for algorithms that can learn with greater autonomy and suggesting a path towards the development of such methods.

7.1 RELATED WORK

Prior work has proposed a number of benchmarks for reinforcement learning, which are often either explicitly episodic (Todorov et al., 2012; Beattie et al., 2016; Chevalier-Boisvert et al., 2018), or consist of games that are implicitly episodic after the player dies or completes the game (Bellemare et al., 2013; Silver et al., 2016). In addition, RL benchmarks have been proposed in the episodic setting for studying a number of orthogonal questions, such multi-task learning (Bellemare et al., 2013; Yu et al., 2020b), sequential task learning (Wołczyk et al., 2021), generalization (Cobbe et al., 2020), and multi-agent learning (Samvelyan et al., 2019; Wang et al., 2020). These benchmarks differ from our own in that we propose to study the challenge of autonomy. Among recent benchmarks, the closest to our own is Jelly Bean World (Platanios et al., 2020), which consists of a set of procedural generated gridworld tasks. While this benchmark also considers the non-episodic setting, our work is inspired by the challenge of autonomous learning in robotics, and hence considers an array of manipulation and locomotion tasks. In addition, our work aims to establish a conceptual framework for evaluating prior algorithms in light of the requirement for persistent autonomy.

Enabling embodied agents to learn continually with minimal interventions is a motivation shared by several subtopics of reinforcement learning research. The setting we study in our work shares conceptual similarities with prior work in continual and lifelong learning (Schmidhuber, 1987a; Thrun and Mitchell, 1995; Parisi et al., 2019; Hadsell et al., 2020). In context of reinforcement learning, this work has studied the problem of episodic learning in sequential MDPs (Khetarpal et al., 2020; Rusu et al., 2016b; Kirkpatrick et al., 2017; Fernando et al., 2017; Schwarz et al., 2018; Mendez et al., 2020), where the main objective is forward/backward transfer or learning in non-stationary dynamics (Chandak et al., 2020; Xie et al., 2020; Lu et al., 2020). In contrast, the emphasis of our work is learning in non-episodic settings while literature in continual RL assumes an episodic setting. As we will discuss, learning autonomously without access to oracle resets is a hard problem even when the task-distribution and dynamics are stationary. In a similar vein, unsupervised RL (Gregor et al., 2016; Pong et al., 2019; Eysenbach et al., 2018; Sharma et al., 2019; Campos et al., 2020) also enables skill acquisition in the absence of rewards, reducing human intervention required for designing reward functions. These works are complimentary to our proposal and form interesting future work.

Reset-free RL has been studied by previous works with a focus on safety (Eysenbach et al., 2017), automated and unattended learning in the real world (Han et al., 2015; Zhu et al., 2020a; Gupta et al., 2021b), skill discovery (Xu et al., 2020; Lu et al., 2020), and providing a curriculum (Sharma et al., 2021a). Strategies to learn reset-free behavior include directly learning a backward reset controller (Eysenbach et al., 2017), learning a set of auxillary tasks that can serve as an approximate reset (Ha et al., 2020; Gupta et al., 2021b), or using a novelty seeking reset controller (Zhu et al., 2020a). Complementary to this literature, we aim to develop a set of benchmarks and a framework that allows for this class of algorithms to be studied in a unified way. Instead of proposing new algorithms, our work is focused on developing a set of unified tasks that emphasize and allow us to study algorithms through the lens of autonomy.

7.2 PRELIMINARIES

Consider a Markov Decision Process (MDP) $\mathcal{M} \equiv (\mathcal{S}, \mathcal{A}, p, r, \rho, \gamma)$ (Sutton and Barto, 2018b). Here, $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}_{\geqslant 0}$ denotes the transition dynamics, $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ denotes the reward function, $\rho : \mathcal{S} \mapsto \mathbb{R}_{\geqslant 0}$ denotes the initial state distribution and $\gamma \in [0, 1)$ denotes the discount factor. The objective in reinforcement learning is to maximize $J(\pi) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$ with respect to the policy π, where $s_0 \sim \rho(\cdot)$, $a_t \sim \pi(\cdot \mid s_t)$ and $s_{t+1} \sim p(\cdot \mid s_t, a_t)$. Importantly, the RL framework assumes the ability to sample $s_0 \sim \rho$ arbitrarily. Typical implementations of reinforcement learning algorithms carry out thousands or millions of these trials, implicitly requiring the environment to provide a mechanism to be "reset" to a state $s_0 \sim \rho$ for every trial.

7.3 AUTONOMOUS REINFORCEMENT LEARNING

In this section, we develop a framework for autonomous reinforcement learning (ARL) that formalizes reinforcement learning in settings without extrinsic interventions. We first define a non-episodic training environment where the agent can autonomously interact with its environment in Section 7.3.1, building on the formalism of standard reinforcement learning. We introduce two distinct evaluation settings as visualized in Figure 38: Section 7.3.1.1 discusses the *deployment setting* where the agent will be deployed in a test environment after training, and the goal is to maximize this "deployed" performance. Section 7.3.1.2 discusses the *continuing setting*, where the agent has no separate deployment phase and aims to maximize the reward accumulated over its lifetime. In its most general form, the latter corresponds closely to standard RL, while the former can be interpreted as a kind of transfer learning. As we will discuss in Section 7.3.2, this general framework can be instantiated such that, for different choices of the underlying

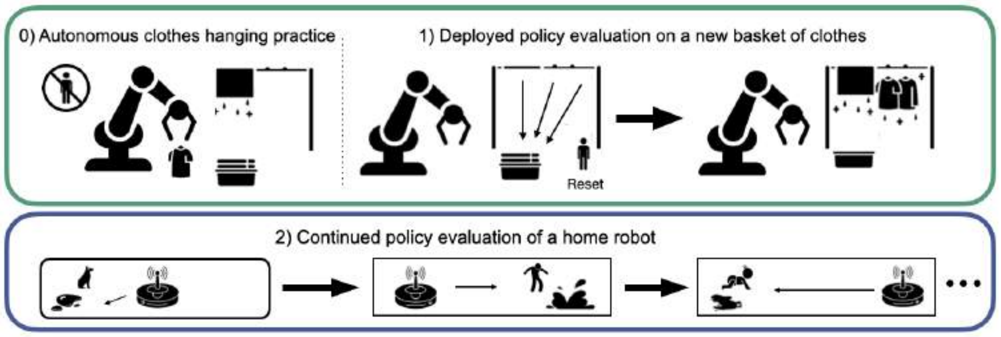

0) Autonomous clothes hanging practice
1) Deployed policy evaluation on a new basket of clothes
Reset
2) Continued policy evaluation of a home robot

Figure 38: Two evaluation schemes in autonomous RL. First, the deployment setting (top row, (1)), where we are interested in obtaining a policy during a training phase, π, that performs well when deployed from a $s_0 \sim \rho$. Second, the continuing setting (bottom row, (2)), where a floor cleaning robot is tasked with keeping a floor clean and is only evaluated on its cumulative performance (Eq. 20) over the agent's lifetime.

MDP, we can model different realistic autonomous RL scenarios such as settings where a robot must learn to reset itself between trials or settings with non-reversible dynamics. Finally, Section 7.3.3 considers algorithm design for autonomous RL, discussing the challenges in autonomous operation while also contrasting the evaluation protocols.

7.3.1 *General Setup*

Our goal is to formalize a problem setting for autonomous reinforcement learning that encapsulates realistic autonomous learning scenarios. We define the setup in terms of a training MDP $\mathcal{M}_T \equiv (\mathcal{S}, \mathcal{A}, p, r, \rho)$, where the environment initializes to $s_0 \sim \rho$, and then the agent interacts with the environment autonomously from then on. Note, this lack of episodic resets in our setup departs not only from the standard RL setting, but from other continual reinforcement learning settings e.g. Wołczyk et al., 2021, where resets are provided between tasks. Symbols retain their meaning from Section 7.2. In this setting, a *learning algorithm* $\mathbb{A}$ can be defined as a function $\mathbb{A} : \{s_i, a_i, s_{i+1}, r_i\}_{i=0}^{t-1} \mapsto (a_t, \pi_t)$, which maps the transitions collected in the environment until the time t (e.g., a replay buffer), to a (potentially exploratory) action $a_t \in \mathcal{A}$ applied in the environment and its best guess at the optimal policy $\pi_t : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}_{\geqslant 0}$ used for *evaluation* at time t. We note that the assumption of a reward function implicitly requires human engineering, but in principle could be relaxed by methods that learn reward functions from data. In addition, we note that a_t does not need to come from π_t, which is already implicit in most reinforcement learning algorithms: Q-learning (Sutton and Barto, 2018b) methods such as DQN (Mnih et al., 2015), DDPG (Lillicrap et al., 2015) use an ϵ-greedy policy as an exploration policy on top of the greedy policy for evaluation. However, our setting necessitates more concerted exploration, and the exploratory action may come from an entirely different policy. Note that the initial state distribution is sampled ($s_0 \sim \rho$) exactly *once* to begin training, and then the algorithm $\mathbb{A}$ is run until $t \to \infty$, generating the sequence $s_0, a_0, s_1, a_1, \ldots$ in the MDP $\mathcal{M}_T$. This is the primary difference compared to the episodic setting described in Section 7.2, which can sample the initial state distribution repeatedly.

7.3.1.1 *ARL Deployment Setting*

Consider the problem where a robot has to learn how to close a door. Traditional reinforcement learning algorithms require several trials, repeatedly requiring interventions to open the door between trials. The desire is that the robot autonomously interacts with the door, learning to open it if it is required to practice closing the door. The output policy of the training procedure is evaluated in its deployment setting, in this case on its ability to close the door. Formally, the evaluation objective J_D for a policy π in the deployment setting can be written as:

$$J_D(\pi) = \mathbb{E}_{s_0 \sim \rho, a_j \sim \pi(\cdot|s_j), s_{j+1} \sim p(\cdot|s_j, a_j)} \Big[\sum_{j=0}^{\infty} \gamma^j r(s_j, a_j) \Big]. \tag{19}$$

Definition 2 (*Deployed Policy Evaluation*). *For an algorithm* $\mathbb{A} : \{s_i, a_i, s_{i+1}\}_{i=0}^{t-1} \mapsto (a_t, \pi_t)$, *deployed policy evaluation* $\mathbb{D}(\mathbb{A})$ *is given by* $\mathbb{D}(\mathbb{A}) = \sum_{t=0}^{\infty} (J_D(\pi^*) - J_D(\pi_t))$, *where* $J_D(\pi)$ *is defined in Eq 19 and* $\pi^* \in \arg\max_\pi J_D(\pi)$.

The evaluation objective $J_D(\pi)$ is identical to the one defined in Section 7.2 on MDP $\mathcal{M}$ (deployment environment). The policy evaluation is "hypothetical", the environment rollouts used for evaluating policies are not used in training. Even though the evaluation trajectories are rolled out from the initial state, there are no interventions in training. Concretely, the algorithmic goal in this setting can be stated as $\min_{\mathbb{A}} \mathbb{D}(\mathbb{A})$. In essence, the policy outputs π_t from the autonomous algorithm $\mathbb{A}$ should match the oracle *deployment* performance, i.e. $J_D(\pi^*)$, as quickly as possible. Note that $J_D(\pi^*)$ is a constant that can be ignored when comparing two algorithms, i.e. we only need to know $J_D(\pi_t)$ for a given algorithm in practice.

7.3.1.2 *ARL Continuing Setting*

For some applications, the agent's experience cannot be separated into training and deployment phases. Agents may have to learn and improve in the environment that they are "deployed" into, and thus these algorithms need to be evaluated on their performance during the agent's lifetime. For example, a robot tasked with keeping the home clean learns and improves on the job as it adapts to the home in which it is deployed. To this end, consider the following definition:

Definition 3 (*Continuing Policy Evaluation*)**.** *For an algorithm* $\mathbb{A} : \{s_i, a_i, s_{t+1}\}_{i=0}^{t-1} \mapsto (a_t, \pi_t)$*, continuing policy evaluation* $\mathbb{C}(\mathbb{A})$ *can be defined as:*

$$\mathbb{C}(\mathbb{A}) = \lim_{h\to\infty} \frac{1}{h} \mathbb{E}_{s_0\sim\rho, a_t\sim\mathbb{A}(\{s_i,a_i,s_{i+1}\}_{i=0}^{t-1}), s_{t+1}\sim p(\cdot|s_t,a_t)} \Big[\sum_{t=0}^{h} r(s_t, a_t)\Big] \tag{20}$$

Here, a_t is the action taken by the algorithm $\mathbb{A}$ based on the transitions collected in the environment until time t, measuring the performance under reward r. The optimization objective can be stated as $\max_{\mathbb{A}} \mathbb{C}(\mathbb{A})$. Note that π_t is not used in computing $\mathbb{C}(\mathbb{A})$. In practice, this amounts to measuring the reward collected by the agent in the MDP $\mathcal{M}_T$ during its lifetime[1].

7.3.2 *How Specific ARL Problems Fit Into Our Framework*

The framework can easily be adapted to model possible autonomous reinforcement learning scenarios that may be encountered:

Intermittent interventions. By default, the agent collects experience in the environment with fully autonomous interaction in the MDP $\mathcal{M}_T$. However, we can model the occasional intervention with transition dynamics defined as $\tilde{p}(\cdot \mid s, a) = (1-\epsilon)p(\cdot \mid s, a) + \epsilon\rho(\cdot)$, where the next state is sampled with $1-\epsilon$ probability from the environment dynamics or with ϵ probability from the initial state distribution via intervention for some $\epsilon \in [0, 1]$. A low ϵ represents very occasional interventions through the training in MDP $\mathcal{M}_T$. In fact, the framework described in Section 7.2, which is predominantly assumed by reinforcement learning algorithms, can be understood to have a large ϵ. To contextualize ϵ,

1 We can also consider the more commonly used setting of expected discounted sum of rewards as the objective. To ensure that future rewards are relevant, the discount factors would need to be much larger than values typically used.

the agent should expect to get an intervention after $1/\epsilon$ steps in the environment. Current episodic settings typically provide an environment reset every 100 to 1000 steps, corresponding to $\epsilon \in (1e\text{-}3, 1e\text{-}2)$ and an autonomous operation time of typically a few seconds to few minutes depending on the environment. While full autonomy would be desirable (that is, $\epsilon = 0$), intervening every few hours to few days may be reasonable to arrange, which corresponds to environment resets every 100,000 to 1,000,000 steps or $\epsilon \in (1e\text{-}6, 1e\text{-}5)$. We evaluate the dependence of algorithms designed for episodic reinforcement learning on the reset frequency in Section 7.5.1.

Irreversible states. An important consideration for developing autonomous algorithms is the "reversibility" of the underlying MDP $\mathcal{M}$. Informally, if the agent can reverse any transition in the environment, the agent is guaranteed to not get stuck in the environment. As an example, a static robot arm can be setup such that there always exists an action sequence to open or close the door. However, the robot arm can push the object out of its reach such that no action sequence can retrieve it. Formally, we require MDPs to be ergodic for them to be considered reversible (Moldovan and Abbeel, 2012). In the case of non-ergodic MDPs, we adapt the ARL framework to enable the agent to request extrinsic interventions, which we discuss in Appendix F.

7.3.3 *Discussion of the ARL Formalism*

The ARL framework provides two evaluation protocols for autonomous RL algorithms. Algorithms can typically optimize only one of the two evaluation metrics. *Which evaluation protocol should the designer optimize for?* In a sense, the need for two evaluation protocols arises from task specific constraints, which themselves can be sometimes relaxed depending on the specific trade-off between the cost of real world training and the cost of intervention. The continuing policy evaluation represents the oracle metric one should strive to optimize when continually operational agents are deployed into dynamic environments. The need for *deployment policy evaluation* arises from two implicit practical constraints: (a) requirement of a large number of trials to solve desired tasks and (b) absence of interventions to enable those trials. If either of these can be easily relaxed, then one could consider optimizing for *continuing policy evaluation*. For example, if the agent can learn in a few trials because it was meta-trained for quick adaptation (Finn et al., 2017c), providing a few interventions for those trials may be reasonable. Similarly, if the interventions are easy to obtain during deployment without incurring significant human cost, perhaps through scripted behaviors or enabled by the deployment setting (for example, sorting trash in a facility), the agent can repeatedly try the task and learn while deployed. However, if these constraints cannot be relaxed at deployment, one should consider optimizing for the *deployment policy evaluation* since this incentivizes the agents to learn targeted behaviors by setting up its own practice problems.

7.4 EARL: ENVIRONMENTS FOR AUTONOMOUS REINFORCEMENT LEARNING

In this section, we introduce the set of environments in our proposed benchmark, **E**nvironments for **A**utonomous **R**einforcement **L**earning (EARL). We first discuss the factors in our design criteria and provide a description of how each environment fits into our overall benchmark philosophy, before presenting the results and analysis. For detailed descriptions of each environment, see Appendix F.

7.4.1 *Benchmark Design Factors*

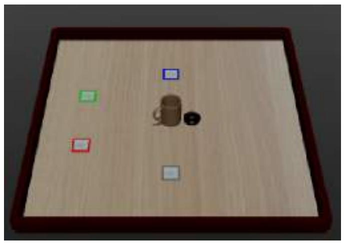
Representative Autonomous Settings. We include a broad array of tasks that reflect the types of autonomous learning scenarios agents may encounter in the real world. This includes different problems in manipulation and locomotion, and tasks with multiple object interactions for which it would be challenging to instrument resets. We also ensure that both the continuing and deployment evaluation protocols of ARL are realistic representative evaluations.

Directed Exploration. In the autonomous setting, the necessity to practice a task again, potentially from different initial states, gives rise to the need for agents to learn rich reset behaviors. For example, in the instance of a robot learning to interact with multiple objects in a kitchen, the robot must also learn to implicitly or explicitly compose different reset behaviors.

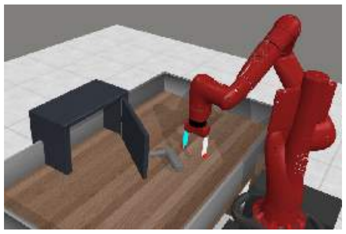
Rewards and Demonstrations. One final design aspect for our benchmark is the choice of reward functions. Dense rewards are a natural choice in certain domains (e.g., locomotion), but designing and providing dense rewards in real world manipulation domains can be exceptionally challenging. Sparse rewards are easier to specify in such scenarios, but this often makes exploration impractical. As a result, prior work has often leveraged demonstrations (e.g., (Gupta et al., 2019)), especially in real world experimentation. To reflect practical usage of RL in real world manipulation settings, we include a small number of demonstrations for the sparse-reward manipulation tasks.

7.4.2 *Environment Descriptions*

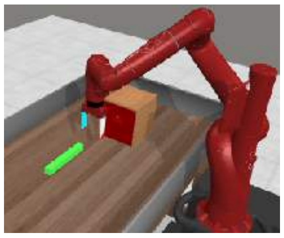
Tabletop-Organization (TO). The Tabletop-Organization task is a diagnostic object manipulation environment proposed by Sharma et al. (2021a). The agent consists of a gripper modeled as a pointmass, which can grasp objects that are close to it. The agent's goal is to bring a mug to four different locations designated by a goal coaster. The agent's reward function is a sparse indicator function when the mug is placed at the goal location. Limited demonstrations are provided to the agent.



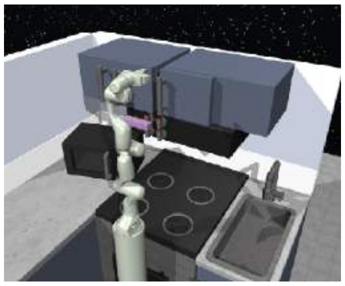
SAWYER-DOOR (SD). The Sawyer-Door task, from the Meta-World benchmark (Yu et al., 2020b) consists of a Sawyer robot arm who's goal is to close the door whenever it is in an open position. The task reward is a sparse indicator function based on the angle of the door. Repeatedly practicing this task implicitly requires the agent to learn to open the door. Limited demonstrations for opening and closing the door are provided.

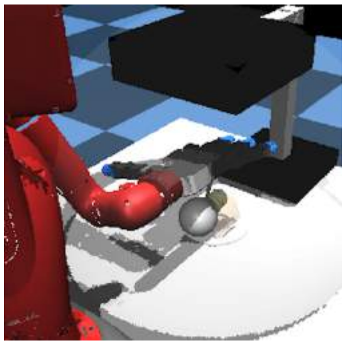


SAWYER-PEG (SP). The Sawyer-Peg task (Yu et al., 2020b) consists of a Sawyer robot required to insert a peg into a designed goal location. The task reward is a sparse indicator function for when the peg is in the goal location. In the deployment setting, the agent must learn to insert the peg starting on the table. Limited demonstrations for inserting and removing the peg are provided.



FRANKA-KITCHEN (FK). The Franka-Kitchen (Gupta et al., 2019) is a domain where a 9-DoF robot, situated in a kitchen environment, is required to solve tasks consisting of compound object interactions. The environment consists of a microwave, a hinged cabinet, a burner, and a slide cabinet. One example task is to open the microwave, door and burner. This domain presents a number of distinct challenges for ARL. First, the compound nature of each task results in a challenging long horizon problem, which introduces exploration and credit assignment challenges. Second, while generalization is important in solving the environment, combining *reset* behaviors are equally important given the compositional nature of the task.





DHAND-LIGHTBULB (DL). The DHand-Lightbulb environment consists of a 22-DoF 4 fingered hand, mounted on a 6 DoF Sawyer robot. The environment is based on one originally proposed by Gupta et al., 2021b. The task in this domain is for the robot to grasp pickup a lightbulb to a specific location. The high-dimensional action space makes the task extremely challenging. In the deployment setting, the bulb can be initialized anywhere on the table, testing the agent on a wide initial state distribution.

MINITAUR-PEN (MP). Finally, the Minitaur-Pen task consists of an 8-DoF Minitaur robot (Coumans and Bai, 2016) confined to a pen environment. The goal of the agent is to navigate to a set of goal locations in the pen. The task is designed to mimic the setup of leaving a robot to learn to navigate within an enclosed setting in an autonomous fashion. This task is different from the other tasks given it is a locomotion task, as opposed to the other tasks being manipulation tasks.

7.5 BENCHMARKING AND ANALYSIS

The aim of this section is to understand the challenges in autonomous reinforcement learning and to evaluate the performance and shortcomings of current autonomous RL algorithms. In Section 7.5.1, we first evaluate standard episodic RL algorithms in ARL settings as they are required to operate with increased autonomy, underscoring the need for a greater focus on autonomy in RL algorithms. We then evaluate prior autonomous learning algorithms on EARL in Section 7.5.2. While these algorithms do improve upon episodic RL methods, they fail to make progress on more challenging tasks compared to methods provided with oracle resets leaving a large gap for improvement. Lastly, in Section 7.5.3, we investigate the learning of existing algorithms, providing a hypothesis for their inadequate performance. We also find that when autonomous RL does succeed, it tends to find more robust policies, suggesting an intriguing connection between autonomy and robustness.

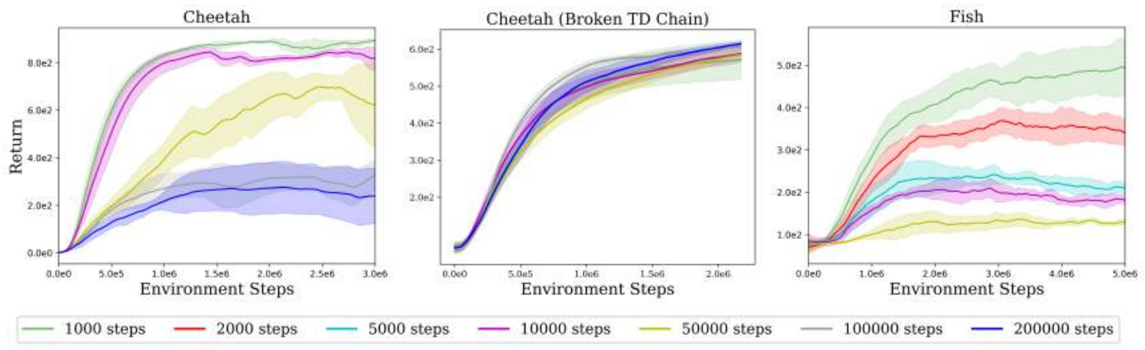


Figure 39: Performance of standard RL at with varying levels of autonomy, ranging from resets provided every 1000 to 200000 steps. Performance degrades substantially as environment resets become infrequent.

7.5.1 *Going from Standard RL to Autonomous RL*

In this section, we evaluate standard RL methods to understand how their performance changes when they are applied naïvely to ARL problems. To create a continuum, we will vary the level of "autonomy" (i.e., frequency of resets), corresponding to ϵ as defined in Section 7.3.2. For these experiments only, we use the simple `cheetah` and `fish` environments from the DeepMind Control Suite (Tassa et al., 2018). We use soft actor-critic (SAC) (Haarnoja et al., 2018b) as a representative standard RL algorithm. We consider different training environments with increasing number of steps between resets, ranging from 1000 to $200,000$ steps. Figure 39 shows the performance of the learned policy as the training progresses, where the return is measured by running the policy for 1000 steps. The `cheetah` environment is an infinite-horizon running environment, so changing the training horizon should not affect the performance theoretically. However, we find that the performance degrades drastically as the training horizon is increased, as shown in Fig 39 (*left*). We attribute this problem to a combination of function approximation and temporal difference learning. Increasing the episode length destabilizes the learning as the effective bootstrapping length increases: the Q-value function $Q^\pi(s_0, a_0)$ bootstraps on the value of $Q^\pi(s_1, a_1)$, which bootstraps on $Q^\pi(s_2, a_2)$ and so on till $Q^\pi(s_{100,000}, a_{100,000})$. To break this chain, we consider a biased TD update: $Q^\pi(s_t, a_t) \leftarrow r(s_t, a_t) + \gamma Q^\pi(s_{t+1}, a_{t+1})$ if t is not a multiple of 1000, else $Q^\pi(s_t, a_t) \leftarrow r(s_t, a_t)$. This is inspired by practical implementations of SAC (Haarnoja et al., 2018d), where the Q-value function regresses to $r(s, a)$ for terminal transitions to stabilize the training. This effectively fixes the problem for `cheetah`, as shown in Figure 39 (*middle*). However, this solution does not translate in general, as can be seen observed in the `fish` environment, where the performance continues to degrade with increasing training horizon, shown in Fig 39 (*right*). The primary difference between `cheetah` and `fish` is that the latter is a goal-reaching domain. The `cheetah` can continue improving its gait on the infinite plane without resets, whereas the `fish` needs to undo the task to practice goal reaching again, creating a non-trivial exploration problem.

7.5.2 *Evaluation: Setup, Metrics, Baselines, and Results*

Training Setup. Every algorithm $\mathbb{A}$ receives a set of initial states $s_0 \sim \rho$, and a set of goals from the goal-distribution $g \sim p(g)$. Demonstrations, if available, are also provided to $\mathbb{A}$. We consider the *intermittent interventions* scenario described in Section 7.3.2 for the benchmark. In practice, we reset the agent after a fixed number of steps H_T. The value of H_T is fixed for every environment, ranging between $100,000$ - $400,000$ steps. Every algorithm $\mathbb{A}$ is run for a fixed number of steps H_{max} after which the training is terminated. Environment-specific details are in Appendix F.0.4.

Evaluation Metrics. For deployed policy evaluation, we compute $\mathbb{D}(\mathbb{A}) = -\sum_{t=0}^{H_{max}} J_D(\pi_t)$, ignoring $J_D(\pi^*)$ as it is a constant for all algorithms. Policy evaluation $J_D(\pi_t)$ is carried out every 10000 training steps, where $J_D(\pi_t) = \sum_{t=0}^{H_E} r(s_t, a_t)$ is the average return accumulated over an episode of H_E steps by running the policy π_t 10 times, starting from $s_0 \sim \rho$ for every trial. These roll-outs are only used for evaluation, and are not provided to the algorithm. In practice, we plot $J_D(\pi_t)$ versus time t, such that minimizing $\mathbb{D}(\mathbb{A})$ can be understood as maximizing the area under the learning curve, which we find more interpretable. Given a finite training budget H_{max}, the policy π_t may be quite suboptimal compared to π^*. Thus, we also report the performance of the final policy, that is $J_D(\pi_{H_{max}})$ in Table 4. For continuing policy evaluation $\mathbb{C}(\mathbb{A})$, we compute the average reward as $\bar{r}(h) = \sum_{t=0}^{h} r(s_t, a_t)/h$. We plot $\bar{r}(h)$ versus h, while we report $\bar{r}(H_{max})$ in Table 5. The evaluation curves corresponding to continuing and deployed policy evaluation are in Appendix **??**.

Baselines. We evaluate forward-backward RL (**FBRL**) (Han et al., 2015; Eysenbach et al., 2017), a perturbation controller (**R3L**) (Zhu et al., 2020a), value-accelerated persistent RL (**VaPRL**) (Sharma et al., 2021a), a comparison to simply running the base RL algorithm with the biased TD update discussed in Section 7.5.1 (**naïve RL**), and finally an oracle (**oracle RL**) where resets are provided are provided every H_E steps (H_T is typically three orders of magnitude larger than H_E). We benchmark VaPRL only when demonstrations are available, in accordance to the proposed algorithm in Sharma et al., 2021a. We average the performance of all algorithms across 5 random seeds. More details

Method	**TO**	**SD**	**SP**	**FK**	**DL**	**MP**
naïve RL	0.32 (0.17)	0.00 (0.00)	0.00 (0.00)	-2705.21 (167.10)	-239.30 (8.85)	-1041.10 (44.58)
FBRL	0.94 (0.04)	**1.00 (0.00)**	0.00 (0.00)	-2733.15 (324.10)	-242.38 (8.84)	-986.34 (67.95)
R3L	0.96 (0.04)	0.54 (0.18)	0.00 (0.00)	-2639.28 (233.28)	728.54 (122.86)	-186.30 (34.79)
VaPRL	**0.98 (0.02)**	0.94 (0.05)	0.02 (0.02)	-	-	-
oracle RL	0.80 (0.11)	**1.00 (0.00)**	**1.00 (0.00)**	**1203.88 (203.86)**	**2028.75 (35.95)**	**-41.50 (3.40)**

Table 4: Average return of the final deployed policy. Performance is averaged over 5 random seeds. The mean and and the standard error are reported, with the best performing entry in bold. For sparse reward domains (**TO, SD, SP**), 1.0 indicates the maximum performance and 0.0 indicates minimum performance.

Method	**TO**	**SD**	**SP**	**FK**	**DL**	**MP**
naïve RL	**0.012 (0.004)**	0.373 (0.073)	< 0.001	**-4.944 (0.440)**	-0.734 (0.024)	-18.193 (2.375)
FBRL	0.005 (0.001)	0.329 (0.080)	**0.003 (0.003)**	-8.754 (0.405)	-0.747 (0.014)	-22.087 (1.285)
R3L	0.001 (0.000)	0.369 (0.016)	< 0.001	-6.577 (0.309)	**0.091 (0.026)**	**-1.093 (0.020)**
VaPRL	0.009 (0.000)	**0.574 (0.085)**	< 0.001	-	-	-

Table 5: Average reward accumulated over the training lifetime in accordance to continuing policy evaluation. Performance is averaged over 5 random seeds. The mean and the standard error (brackets) are reported, with the best performing entry in bold.

pertaining to these algorithms can be found in Appendix F.0.3, F.0.5.

Overall, we see in Table 4 that based on the deployed performance of the final policy, autonomous RL algorithms substantially underperform oracle RL and fail to make any progress on `sawyer-peg` and `franka-kitchen`. Notable exceptions are the performances of VaPRL on `tabletop-organization` and R3L on `minitaur-pen`, outperforming oracle RL. Amongst the autonomous RL algorithms, VaPRL does best when the demonstrations are given and R3L does well on domains when no demonstrations are provided. Nonetheless, this leaves substantial room for improvement for future works evaluating on this benchmark. More detailed learning curves are shown in Section **??**. In the continuing setting, we find that naïve RL performs well on certain domains (best on 2 out of 6 domains). This is unsurprising, as naïve RL is incentivized to occupy the final "goal" position, and continue to accumulate over the course of its lifetime, whereas other algorithms are explicitly incentivized to explore. Perhaps surprisingly, we find that VaPRL on `sawyer-door` and R3L on `dhand-lightbulb` and `minitaur` does better than naïve RL, suggesting that optimizing for deployed performance can also improve the continuing performance. Overall, we find that performance in the continuing setting does not necessarily translate to improved performance in the deployed policy evaluation, emphasizing the differences between these two evaluation schemes.

7.5.3 *Analyzing Autonomous RL Algorithms*

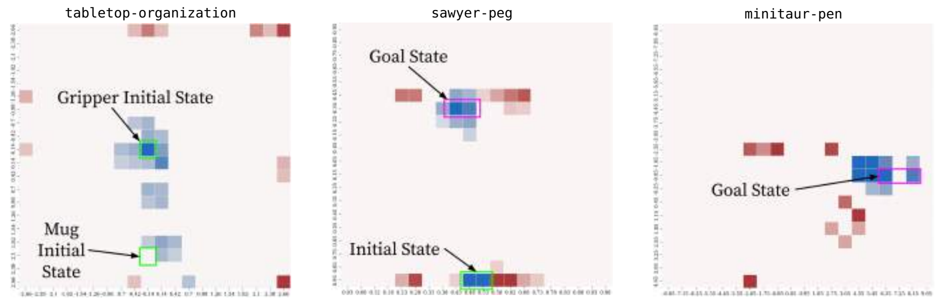


Figure 40: Comparing the distribution of states visited with resets (*blue*) and without resets (*brown*). Heatmaps visualize the difference between state visitations for oracle RL and FBRL, thresholded to highlight states with large differences. Resets enable the agent to stay around the initial state distribution and the goal distribution, whereas the agents operating autonomously skew farther away, posing an exploration challenge.

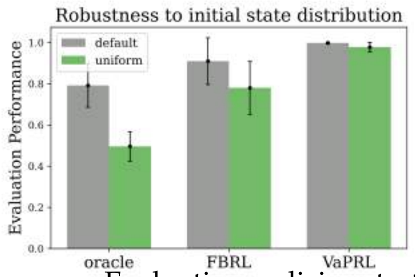


Figure 41: Evaluating policies starting from a uniform initial state distribution. Policies learned via autonomous RL (FBRL and VaPRL) are more robust to initial state distribution than policies learned in oracle RL.

A hypothesis to account for the relative underperformance of autonomous RL algorithms compared to oracle RL is that environment resets constrain the state distribution visited by the agent close to the initial and the goal states. When operating autonomously for long periods of time, the agent can skew far away from the goal states, creating a hard exploration challenge. To test this hypothesis, we compare the state distribution when using oracle RL (*in blue*) versus FBRL (*in brown*) in Figure 40. We visualize the (x, y) positions visited by the gripper for `tabletop-organization`, the (x, y) positions of the peg for `sawyer-peg` and the x, y positions of the minitaur for `minitaur-pen`. As seen in the figure, autonomous operation skews the gripper towards the corners in `tabletop-organization`, the peg is stuck around the goal box and minitaur can completely go away from the goal distribution. However, when autonomous algorithms are able to solve the task, the learned policies can be more robust as they are faced with a tougher exploration challenge during training. We visualize this in Figure 41, where we test the final policies learned by oracle RL, FBRL and VaPRL on `tabletop-organization` starting from a uniform state distribution instead of the default ones. We observe that the policies learned by VaPRL and FBRL depreciate by 2% and

14.3% respectively, which is much smaller than the 37.4% depreciation of the policy learned by oracle RL, suggesting that autonomous RL can lead to more robust policies.

7.6 CONCLUSION

We proposed a formalism and benchmark for autonomous reinforcement learning, including an evaluation of prior state-of-the algorithms with explicit emphasis on autonomy. We present two distinct evaluation settings, which represent different practical use cases for autonomous learning. The main conclusion from our experiments is that existing algorithms generally do not perform well in scenarios that demand autonomy during learning. We also find that exploration challenges, while present in the episodic setting, are greatly exacerbated in the autonomous setting.

While our work focuses on predominantly autonomous settings, there may be task-specific trade-offs between learning speed and the cost of human interventions, and it may indeed be beneficial to provide some human supervision to curtail total training time. How to best provide this supervision (rewards and goal setting, demonstrations, resets etc) while minimizing human cost provides a number of interesting directions for future work. However, we believe that there is a lot of room to improve the autonomous learning algorithms and our work attempts to highlight the importance and challenge of doing so.

8

CONCLUSION & FUTURE DIRECTIONS

In this thesis, we have considered two important challenges in building artificial embodied agents. We began by first considering the problem of learning efficiently under limited supervision via meta-learning, in the problem of reward specification via inverse reinforcement learning (Chapter 2). We showed that by leveraging algorithms designed for fast adaptation, we could make use of shared structure between tasks in order to rapidly infer reward functions. Next, in Chapter 3, we turned to the problem of dealing with uncertainty that is inevitable when learning from sparse supervision. We showed that by developing a probabilistic formulation of gradient based meta-learning, we could generate hypotheses in new classification settings, which could be leveraged to give more calibrated uncertainty measures.

In order to enable the techniques used in the first part of this thesis, we crucially assumed access to a dataset of related "tasks". In the applications we considered, we assume that a human had curated such a dataset. In the second part of this thesis, we considered the natural question of how an agent would go about continually solving new tasks in a never ending lifelong way. In Chapters 4, we started by examining how to extend the reinforcement learning setting to allow the agent to continually learn skills in an intervention-free manner. Next, we showed in Chapter 5 and 6 how some of these ideas could be transferred to the real world in the challenging problem setting of dexterous manipulation, where the demonstrated on a real robot the acquisition of multiple complex behaviors from continual interaction. Finally, in Chapter 7, we proposed a formalism to study the problem setting of reset-free RL, which we call "Autonomous RL", and provide a benchmark and analysis for future work and research.

A central thread of this thesis work has been to minimize the amount of human effort needed to enable embodied learning systems. While this thesis has attempted to bring us closer to the vision of embodied systems learning in the wild, towards the goal of autonomous embodied systems there remains many challenges. Below we discuss some potential directions and open questions.

Automating Task Construction: While the work in Chapters 4, 5, 6 took a step in the direction of allowing tasks to be solved in a more natural online manner, one of the key sources of human supervision is the necessary construction of a set of "tasks". In Chapter 6, we looked at how some of these tasks could be specified efficiently by humans, one more scalable avenue would be to leverage a large knowledge base of "steps" that could be used to break down a task automatically for an agent. The key design question here would be to design "what" knowledge base would be appropriate. For example, some recent work has explored querying large language models pretrained on large text datasets (Huang et al., 2022) for planning. Combined with the appropriate mechanism to ground the "steps", this in principle could dramatically improve an agent's ability to plan a set of subgoals to solve a problem in a reset-free manner.

Where does prior experience fit into the picture of lifelong learning? We have shown in this work the potential to acquire reset-free behavior from autonomous data collection. In this work however, we have done so starting each of the agents from scratch. In many real world robotics problems however, we often have access to large datasets of prior experience that can be used to guide exploration or bootstrap learning. The field of leveraging offline data for control (sometimes called offline RL (Levine et al., 2020)) has shown tremendous progress in recent years. One simple question with an abundance of design decisions would be to investigate how best to reuse such experience, and if the datasets needed for autonomous RL are different given the particular challenges (e.g., in exploration) of the setting.

How do we ensure safety in an embodied learning system? Finally, one very important question we did not consider in this work is the question of safety. Along the critical path to having agents autonomously explore the world will undoubtedly involve ensuring that they do not harm themselves or others. How best to answer this question would likely need to be somewhat problem specific, as certain domains are much more safety critical than others (e.g., surgical settings, autonomous driving), but could potentially include ingredients such as prior data as a means to communicate "dangerous states".

BIBLIOGRAPHY

Abbeel, Pieter and Andrew Y. Ng (2004). "Apprenticeship Learning via Inverse Reinforcement Learning." In: *Proceedings of the Twenty-first International Conference on Machine Learning*. ICML. New York, NY, USA (cit. on p. 6).

Agrawal, Pulkit, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine (2016). "Learning to Poke by Poking: Experiential Learning of Intuitive Physics." In: *CoRR* abs/1606.07419. arXiv: 1606.07419. URL: http://arxiv.org/abs/1606.07419 (cit. on p. 52).

Ahn, Michael, Henry Zhu, Kristian Hartikainen, Hugo Ponte, Abhishek Gupta, Sergey Levine, and Vikash Kumar (2020). "ROBEL: RObotics BEnchmarks for Learning with low-cost robots." In: *Conference on Robot Learning*. PMLR, pp. 1300–1313 (cit. on pp. 52, 54).

Akgun, Baris, Maya Cakmak, Jae Wook Yoo, and Andrea Lockerd Thomaz (2012). "Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective." In: *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pp. 391–398 (cit. on p. 72).

Akinola, Iretiayo, Jacob Varley, and Dmitry Kalashnikov (2020). "Learning precise 3d manipulation from multiple uncalibrated cameras." In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 4616–4622 (cit. on p. 71).

Alet, Ferran, Tomás Lozano-Pérez, and Leslie P Kaelbling (2018). "Modular meta-learning." In: *arXiv preprint arXiv:1806.10166* (cit. on p. 8).

Allshire, Arthur, Mayank Mittal, Varun Lodaya, Viktor Makoviychuk, Denys Makoviichuk, Felix Widmaier, Manuel Wüthrich, Stefan Bauer, Ankur Handa, and Animesh Garg (2021). "Transferring dexterous manipulation from gpu simulation to a remote real-world trifinger." In: *arXiv preprint arXiv:2108.09779* (cit. on p. 71).

Amodei, Dario, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané (2016). "Concrete problems in AI safety." In: *arXiv preprint arXiv:1606.06565* (cit. on p. 5).

Andrychowicz, Marcin, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas (2016). "Learning to learn by gradient descent by gradient descent." In: *Advances in Neural Information Processing Systems*, pp. 3981–3989 (cit. on pp. 8, 10).

Argall, Brenna D, Sonia Chernova, Manuela Veloso, and Brett Browning (2009). "A survey of robot learning from demonstration." In: *Robotics and autonomous systems* 57.5, pp. 469–483 (cit. on p. 72).

Babeş-Vroman, Monica, Vukosi Marivate, Kaushik Subramanian, and Michael Littman (2011). "Apprenticeship Learning About Multiple Intentions." In: *International Conference on International Conference on Machine Learning*. ICML. USA (cit. on p. 8).

Bacon, Pierre-Luc, Jean Harb, and Doina Precup (2017). "The option-critic architecture." In: *Thirty-First AAAI Conference on Artificial Intelligence* (cit. on p. 41).

Bai, Yunfei and C Karen Liu (2014). "Dexterous manipulation using both palm and fingers." In: *ICRA 2014*. IEEE (cit. on p. 54).

Baier-Löwenstein, Tim and Jianwei Zhang (2007). "Learning to grasp everyday objects using reinforcement-learning with automatic value cut-off." In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 29 - November 2, 2007, Sheraton Hotel and Marina, San Diego, California, USA*. IEEE, pp. 1551–1556. DOI: 10.1109/IROS.2007.4399053. URL: https://doi.org/10.1109/IROS.2007.4399053 (cit. on p. 54).

Baker, Bowen, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch (2019). "Emergent tool use from multi-agent autocurricula." In: *arXiv preprint arXiv:1909.07528* (cit. on p. 41).

Baker, Chris, Joshua B Tenenbaum, and Rebecca R Saxe (Jan. 2007). "Goal inference as inverse planning." In: (cit. on p. 6).

Baranes, Adrien and Pierre-Yves Oudeyer (2013). "Active learning of inverse models with intrinsically motivated goal exploration in robots." In: *Robotics and Autonomous Systems* 61.1, pp. 49–73 (cit. on p. 40).

Barber, David and Christopher M Bishop (1998). "Ensemble learning for multi-layer networks." In: *neural information processing systems (NIPS)* (cit. on p. 25).

Barto, Andrew G (2013). "Intrinsic motivation and reinforcement learning." In: *Intrinsically motivated learning in natural and artificial systems*. Springer, pp. 17–47 (cit. on p. 40).

Beattie, Charles, Joel Z. Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Victor Valdés, Amir Sadik, Julian Schrittwieser, Keith Anderson, Sarah York, Max Cant, Adam Cain, Adrian Bolton, Stephen Gaffney, Helen King, Demis Hassabis, Shane Legg, and Stig. Petersen (2016). "DeepMind Lab." In: *arXiv:1612.03801 [cs.AI]* (cit. on p. 85).

Bellemare, Marc G, Yavar Naddaf, Joel Veness, and Michael Bowling (2013). "The arcade learning environment: An evaluation platform for general agents." In: *Journal of Artificial Intelligence Research* 47, pp. 253–279 (cit. on p. 85).

Bellemare, Marc, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos (2016). "Unifying count-based exploration and intrinsic motivation." In: *Advances in Neural Information Processing Systems*, pp. 1471–1479 (cit. on p. 40).

Bengio, Yoshua, Samy Bengio, and Jocelyn Cloutier. *Learning a synaptic learning rule* (cit. on pp. 3, 8).

Blundell, Charles, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra (2015). "Weight uncertainty in neural networks." In: *arXiv preprint arXiv:1505.05424* (cit. on p. 25).

Brant, Jonathan C and Kenneth O Stanley (2017). "Minimal criterion coevolution: a new approach to open-ended search." In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 67–74 (cit. on p. 41).

Braun, Daniel A, Carsten Mehring, and Daniel M Wolpert (2010). "Structure learning in action." In: *Behavioural brain research* (cit. on p. 23).

Brown, Daniel S, Wonjoon Goo, and Scott Niekum (2020). "Better-than-demonstrator imitation learning via automatically-ranked demonstrations." In: *Conference on robot learning*. PMLR, pp. 330–359 (cit. on p. 72).

Burda, Yuri, Harrison Edwards, Amos J. Storkey, and Oleg Klimov (2019). "Exploration by random network distillation." In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. URL: https://openreview.net/forum?id=H1lJJnR5Ym (cit. on pp. 65, 73, 81).

Burda, Yuri, Harrison Edwards, Amos Storkey, and Oleg Klimov (2018). "Exploration by random network distillation." In: *arXiv preprint arXiv:1810.12894* (cit. on pp. 48, 145, 153).

Calandra, Roberto, André Seyfarth, Jan Peters, and Marc Peter Deisenroth (2016). "Bayesian optimization for learning gaits under uncertainty - An experimental comparison on a dynamic bipedal walker." In: *Ann. Math. Artif. Intell.* 76.1-2, pp. 5–23. DOI: 10.1007/s10472-015-9463-9. URL: https://doi.org/10.1007/s10472-015-9463-9 (cit. on p. 54).

Campos, Víctor, Alexander Trott, Caiming Xiong, Richard Socher, Xavier Giró-i Nieto, and Jordi Torres (2020). "Explore, discover and learn: Unsupervised discovery of state-covering skills." In: *International Conference on Machine Learning*. PMLR, pp. 1317–1327 (cit. on p. 85).

Chandak, Yash, Georgios Theocharous, Shiv Shankar, Martha White, Sridhar Mahadevan, and Philip Thomas (2020). “Optimizing for the future in non-stationary MDPs.” In: *International Conference on Machine Learning*. PMLR, pp. 1414–1425 (cit. on p. 85).

Chatzilygeroudis, Konstantinos, Vassilis Vassiliades, and Jean-Baptiste Mouret (2018). “Reset-free trial-and-error learning for robot damage recovery.” In: *Robotics and Autonomous Systems* 100, pp. 236–250 (cit. on p. 40).

Chebotar, Yevgen, Karol Hausman, Marvin Zhang, Gaurav Sukhatme, Stefan Schaal, and Sergey Levine (2017). “Combining model-based and model-free updates for trajectory-centric reinforcement learning.” In: *International conference on machine learning*. PMLR, pp. 703–711 (cit. on pp. 2, 83).

Chebotar, Yevgen, Mrinal Kalakrishnan, Ali Yahya, Adrian Li, Stefan Schaal, and Sergey Levine (2016). “Path Integral Guided Policy Search.” In: *CoRR* abs/1610.00529. arXiv: 1610.00529. URL: http://arxiv.org/abs/1610.00529 (cit. on pp. 52, 54).

Chentanez, Nuttapong, Andrew G Barto, and Satinder P Singh (2005). “Intrinsically motivated reinforcement learning.” In: *Advances in neural information processing systems*, pp. 1281–1288 (cit. on p. 40).

Chevalier-Boisvert, Maxime, Lucas Willems, and Suman Pal (2018). *Minimalistic Gridworld Environment for OpenAI Gym*. https://github.com/maximecb/gym-minigrid (cit. on p. 85).

Choi, Changhyun, Wilko Schwarting, Joseph DelPreto, and Daniela Rus (2018). “Learning object grasping for soft robot hands.” In: *IEEE Robotics and Automation Letters* 3.3, pp. 2370–2377 (cit. on p. 54).

Choi, Jaedeug and Kee-Eung Kim (2012). “Nonparametric Bayesian Inverse Reinforcement Learning for Multiple Reward Functions.” In: *Advances in Neural Information Processing Systems*. NIPS. USA (cit. on p. 7).

Co-Reyes, John D., Abhishek Gupta, Suvansh Sanjeev, Nick Altieri, John DeNero, Pieter Abbeel, and Sergey Levine (2018). “Guiding Policies with Language via Meta-Learning.” In: *CoRR* abs/1811.07882. arXiv: 1811.07882. URL: http://arxiv.org/abs/1811.07882 (cit. on p. 72).

Co-Reyes, John D, Suvansh Sanjeev, Glen Berseth, Abhishek Gupta, and Sergey Levine (2020). “Ecological Reinforcement Learning.” In: *arXiv preprint arXiv:2006.12478* (cit. on p. 83).

Cobbe, Karl, Chris Hesse, Jacob Hilton, and John Schulman (2020). “Leveraging procedural generation to benchmark reinforcement learning.” In: *International conference on machine learning*. PMLR, pp. 2048–2056 (cit. on p. 85).

Colson, Benoît, Patrice Marcotte, and Gilles Savard (2007). “An overview of bilevel optimization.” In: *Annals of operations research* 153.1, pp. 235–256 (cit. on p. 44).

Coumans, Erwin and Yunfei Bai (2016). *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. `http://pybullet.org` (cit. on pp. 84, 93).

Cui, Yuchen and Scott Niekum (2018). “Active reward learning from critiques.” In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, pp. 6907–6914 (cit. on p. 72).

Daumé III, Hal (2009). “Bayesian multitask learning with latent hierarchies.” In: *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, pp. 135–142 (cit. on p. 24).

Dayan, Peter and Geoffrey E Hinton (1993). “Feudal reinforcement learning.” In: *Advances in neural information processing systems*, pp. 271–278 (cit. on pp. 38, 41).

Deimel, Raphael and Oliver Brock (2016). “A novel type of compliant and underactuated robotic hand for dexterous grasping.” In: *The International Journal of Robotics Research* 35.1-3, pp. 161–185 (cit. on p. 71).

Deisenroth, Marc Peter, Peter Englert, Jan Peters, and Dieter Fox (2014). “Multi-task policy search for robotics.” In: *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*. IEEE, pp. 3876–3881. DOI: `10.1109/ICRA.2014.6907421`. URL: `https://doi.org/10.1109/ICRA.2014.6907421` (cit. on p. 55).

Dietterich, Thomas G (2000). “Hierarchical reinforcement learning with the MAXQ value function decomposition.” In: *Journal of artificial intelligence research* 13, pp. 227–303 (cit. on pp. 38, 41).

Dimitrakakis, Christos and Constantin A. Rothkopf (2012). “Bayesian Multitask Inverse Reinforcement Learning.” In: *European Conference on Recent Advances in Reinforcement Learning*. EWRL (cit. on p. 7).

Duan, Yan, Marcin Andrychowicz, Bradly C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba (2017). “One-Shot Imitation Learning.” In: *NIPS*, pp. 1087–1098 (cit. on p. 15).

Duan, Yan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel (2016). “RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning.” In: *arXiv preprint arXiv:1611.02779* (cit. on pp. 8, 10, 25).

Duvenaud, David, Dougal Maclaurin, and Ryan Adams (2016). “Early stopping as nonparametric variational inference.” In: *Artificial Intelligence and Statistics*, pp. 1070–1077 (cit. on p. 14).

Edwards, Harrison and Amos Storkey (2017). "Towards a Neural Statistician." In: *International Conference on Learning Representations (ICLR)*. Toulon, France (cit. on p. 25).

Eysenbach, Benjamin, Shixiang Gu, Julian Ibarz, and Sergey Levine (2017). "Leave no trace: Learning to reset for safe and autonomous reinforcement learning." In: *arXiv preprint arXiv:1711.06782* (cit. on pp. 38, 40, 48, 49, 52, 54, 58, 65, 66, 72, 74, 83, 85, 86, 95, 145, 153, 167, 175).

Eysenbach, Benjamin, Abhishek Gupta, Julian Ibarz, and Sergey Levine (2018). "Diversity is all you need: Learning skills without a reward function." In: *arXiv preprint arXiv:1802.06070* (cit. on pp. 40, 42, 46, 48, 51, 85, 141, 144, 149, 152).

Fei-Fei, Li et al. (2003). "A Bayesian approach to unsupervised one-shot learning of object categories." In: *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, pp. 1134–1141 (cit. on p. 24).

Fernando, Chrisantha, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra (2017). "Pathnet: Evolution channels gradient descent in super neural networks." In: *arXiv preprint arXiv:1701.08734* (cit. on p. 85).

Fiez, Tanner, Benjamin Chasnov, and Lillian J Ratliff (2019). "Convergence of learning dynamics in stackelberg games." In: *arXiv preprint arXiv:1906.01217* (cit. on pp. 44, 47).

Finn, Chelsea, Pieter Abbeel, and Sergey Levine (2017a). "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks." In: *International Conference on Machine Learning (ICML)*. Sydney, Australia (cit. on pp. 3, 24–26, 29, 139).

Finn, Chelsea, Pieter Abbeel, and Sergey Levine (2017b). "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks." In: *International Conference on Machine Learning* (cit. on pp. 7, 8, 10).

Finn, Chelsea, Pieter Abbeel, and Sergey Levine (2017c). "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks." In: *International conference on machine learning*. PMLR (cit. on p. 90).

Finn, Chelsea and Sergey Levine (2017). "Meta-Learning and Universality: Deep Representations and Gradient Descent can Approximate any Learning Algorithm." In: *arXiv preprint arXiv:1710.11622* (cit. on pp. 24, 140).

Finn, Chelsea, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine (2017d). "One-shot visual imitation learning via meta-learning." In: *arXiv preprint arXiv:1709.04905* (cit. on pp. 8, 15, 129, 139).

Florensa, Carlos, Jonas Degrave, Nicolas Heess, Jost Tobias Springenberg, and Martin Riedmiller (2019). “Self-supervised learning of image embedding for continuous control.” In: *arXiv preprint arXiv:1901.00943* (cit. on p. 41).

Florensa, Carlos, Yan Duan, and Pieter Abbeel (2017). “Stochastic neural networks for hierarchical reinforcement learning.” In: *arXiv preprint arXiv:1704.03012* (cit. on pp. 40, 41).

Fortunato, Meire, Charles Blundell, and Oriol Vinyals (2017). “Bayesian recurrent neural networks.” In: *arXiv preprint arXiv:1704.02798* (cit. on p. 28).

Fu, Justin, Anoop Korattikara, Sergey Levine, and Sergio Guadarrama (2019). “From Language to Goals: Inverse Reinforcement Learning for Vision-Based Instruction Following.” In: *International Conference on Learning Representations (ICLR)* (cit. on pp. 16–19, 132).

Fu, Justin, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine (2020). “D4RL: Datasets for Deep Data-Driven Reinforcement Learning.” In: *https://arxiv.org/abs/2004.07219* (cit. on pp. 142, 150).

Fu, Justin, Katie Luo, and Sergey Levine (2018a). “Learning Robust Rewards with Adverserial Inverse Reinforcement Learning.” In: *International Conference on Learning Representations* (cit. on p. 7).

Fu, Justin, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine (2018b). “Variational inverse control with events: A general framework for data-driven reward definition.” In: *arXiv preprint arXiv:1805.11686* (cit. on pp. 72, 77, 78, 145, 153, 162, 163).

Furukawa, Noriatsu, Akio Namiki, Senoo Taku, and Masatoshi Ishikawa (2006). “Dynamic regrasping using a high-speed multifingered hand and a high-speed vision system.” In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, pp. 181–187 (cit. on p. 54).

Gao, Jing, Wei Fan, Jing Jiang, and Jiawei Han (2008). “Knowledge transfer via multiple model local structure mapping.” In: *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. ACM (cit. on p. 24).

Garcıa, Javier and Fernando Fernández (2015). “A comprehensive survey on safe reinforcement learning.” In: *Journal of Machine Learning Research* 16.1, pp. 1437–1480 (cit. on p. 75).

Garcia, Victor and Joan Bruna (2017). “Few-Shot Learning with Graph Neural Networks.” In: *arXiv preprint arXiv:1711.04043* (cit. on p. 140).

Ghadirzadeh, Ali, Atsuto Maki, Danica Kragic, and Mårten Björkman (2017a). “Deep Predictive Policy Training using Reinforcement Learning.” In: *CoRR* abs/1703.00727. arXiv: `1703.00727`. URL: `http://arxiv.org/abs/1703.00727` (cit. on p. 52).

Ghadirzadeh, Ali, Atsuto Maki, Danica Kragic, and Mårten Björkman (2017b). "Deep predictive policy training using reinforcement learning." In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 2351–2358 (cit. on pp. 2, 83).

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks." In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256 (cit. on pp. 128, 131).

Grant, Erin, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths (2018a). "Recasting Gradient-Based Meta-Learning as Hierarchical Bayes." In: *International Conference on Learning Representations (ICLR)* (cit. on pp. 13, 14).

Grant, Erin, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths (2018b). "Recasting Gradient-Based Meta-Learning as Hierarchical Bayes." In: *International Conference on Learning Representations (ICLR)* (cit. on pp. 25, 27, 29, 140).

Grant, Erin, Chelsea Finn, Josh Peterson, Josh Abbott, Sergey Levine, Trevor Darrell, and Thomas Griffiths (2017). "Concept Acquisition through Meta-Learning." In: *NIPS Workshop on Cognitively Informed Artificial Intelligence* (cit. on p. 34).

Graves, Alex (2011). "Practical variational inference for neural networks." In: *Neural Information Processing Systems (NIPS)* (cit. on p. 25).

Gregor, Karol, Danilo Jimenez Rezende, and Daan Wierstra (2016). "Variational intrinsic control." In: *arXiv preprint arXiv:1611.07507* (cit. on pp. 38, 40, 42, 85).

Gu, Shixiang, Ethan Holly, Timothy P. Lillicrap, and Sergey Levine (2016). "Deep Reinforcement Learning for Robotic Manipulation." In: *CoRR* abs/1610.00633. arXiv: `1610.00633`. URL: `http://arxiv.org/abs/1610.00633` (cit. on p. 54).

Gupta, Abhishek, Clemens Eppner, Sergey Levine, and Pieter Abbeel (2016a). "Learning dexterous manipulation for a soft robotic hand from human demonstrations." In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2016, Daejeon, South Korea, October 9-14, 2016*, pp. 3786–3793. DOI: `10.1109/IROS.2016.7759557`. URL: `https://doi.org/10.1109/IROS.2016.7759557` (cit. on p. 54).

Gupta, Abhishek, Clemens Eppner, Sergey Levine, and Pieter Abbeel (2016b). "Learning dexterous manipulation for a soft robotic hand from human demonstrations." In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3786–3793 (cit. on p. 71).

Gupta, Abhishek, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman (2019). "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning." In: *arXiv preprint arXiv:1910.11956* (cit. on pp. 84, 91, 92, 176).

Gupta, Abhishek, Justin Yu, Tony Z Zhao, Vikash Kumar, Aaron Rovinsky, Kelvin Xu, Thomas Devlin, and Sergey Levine (2021a). "Reset-Free Reinforcement Learning via Multi-Task Learning: Learning Dexterous Manipulation Behaviors without Human Intervention." In: *arXiv preprint arXiv:2104.11203* (cit. on p. 4).

Gupta, Abhishek, Justin Yu, Tony Z Zhao, Vikash Kumar, Aaron Rovinsky, Kelvin Xu, Thomas Devlin, and Sergey Levine (2021b). "Reset-Free Reinforcement Learning via Multi-Task Learning: Learning Dexterous Manipulation Behaviors without Human Intervention." In: *arXiv preprint arXiv:2104.11203* (cit. on pp. 73, 80, 82–86, 93, 175).

Ha, Sehoon, Peng Xu, Zhenyu Tan, Sergey Levine, and Jie Tan (2020). *Learning to Walk in the Real World with Minimal Human Effort*. arXiv: 2002.08550 [cs.RO] (cit. on pp. 54, 55, 86).

Haarnoja, Tuomas, Vitchyr Pong, Aurick Zhou, Murtaza Dalal, Pieter Abbeel, and Sergey Levine (2018a). "Composable deep reinforcement learning for robotic manipulation." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 6244–6251 (cit. on p. 42).

Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel, and Sergey Levine (2018b). "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor." In: *International conference on machine learning*. PMLR, pp. 1861–1870 (cit. on p. 94).

Haarnoja, Tuomas, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. (2018c). "Soft actor-critic algorithms and applications." In: *arXiv preprint arXiv:1812.05905* (cit. on pp. 47, 48, 59, 65, 78, 80, 163).

Haarnoja, Tuomas, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. (2018d). "Soft actor-critic algorithms and applications." In: *arXiv preprint arXiv:1812.05905* (cit. on pp. 94, 177).

Hadsell, Raia, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu (2020). "Embracing Change: Continual Learning in Deep Neural Networks." In: *Trends in Cognitive Sciences* (cit. on p. 85).

Han, Weiqiao, Sergey Levine, and Pieter Abbeel (2015). "Learning compound multi-step controllers under unknown dynamics." In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 6435–6442 (cit. on pp. 40, 54, 58, 72, 74, 84, 85, 95, 175).

Hausman, Karol, Yevgen Chebotar, Stefan Schaal, Gaurav Sukhatme, and Joseph J Lim (2017). "Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets." In: *Advances in Neural Information Processing Systems*, pp. 1235–1245 (cit. on p. 7).

Heess, Nicolas, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin A. Riedmiller, and David Silver (2017). "Emergence of Locomotion Behaviours in Rich Environments." In: *CoRR* abs/1707.02286. arXiv: `1707.02286`. URL: `http://arxiv.org/abs/1707.02286` (cit. on p. 54).

Higgins, Irina, Loic Matthey, Xavier Glorot, Arka Pal, Benigno Uria, Charles Blundell, Shakir Mohamed, and Alexander Lerchner (2017). "Early visual concept learning with unsupervised deep learning." In: *International Conference on Learning Representations (ICLR)* (cit. on p. 35).

Hinton, Geoffrey E and Drew Van Camp (1993). "Keeping the neural networks simple by minimizing the description length of the weights." In: *Conference on Computational learning theory* (cit. on p. 25).

Hinton, Geoffrey, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. (2012). "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups." In: *IEEE Signal processing magazine* 29.6, pp. 82–97 (cit. on p. 2).

Hiraoka, Takuya, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka (2021). "Dropout Q-Functions for Doubly Efficient Reinforcement Learning." In: *arXiv preprint arXiv:2110.02034* (cit. on pp. 77, 163).

Ho, Jonathan and Stefano Ermon (2016). "Generative Adversarial Imitation Learning." In: *Neural Information Processing Systems (NIPS)* (cit. on p. 7).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory." In: *Neural computation* 9.8, pp. 1735–1780 (cit. on p. 128).

Hochreiter, Sepp, A Younger, and Peter Conwell (2001). "Learning to learn using gradient descent." In: *International Conference on Artificial Neural Networks (ICANN)* (cit. on p. 25).

Hoffman, Matthew D, David M Blei, Chong Wang, and John Paisley (2013). "Stochastic variational inference." In: *The Journal of Machine Learning Research* (cit. on p. 25).

Hoof, Herke van, Tucker Hermans, Gerhard Neumann, and Jan Peters (2015a). "Learning robot in-hand manipulation with tactile features." In: *Humanoid Robots (Humanoids)*. IEEE (cit. on p. 54).

Hoof, Herke van, Tucker Hermans, Gerhard Neumann, and Jan Peters (2015b). "Learning robot in-hand manipulation with tactile features." In: *15th IEEE-RAS International*

Conference on Humanoid Robots, Humanoids 2015, Seoul, South Korea, November 3-5, 2015. IEEE, pp. 121–127. DOI: 10.1109/HUMANOIDS.2015.7363524. URL: https://doi.org/10.1109/HUMANOIDS.2015.7363524 (cit. on p. 54).

Huang, Wenlong, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. (2022). "Inner Monologue: Embodied Reasoning through Planning with Language Models." In: *arXiv preprint arXiv:2207.05608* (cit. on p. 100).

Jain, Divye, Andrew Li, Shivam Singhal, Aravind Rajeswaran, Vikash Kumar, and Emanuel Todorov (2019). "Learning deep visuomotor policies for dexterous hand manipulation." In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3636–3643 (cit. on pp. 54, 71).

James, Stephen, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison (2019). *RLBench: The Robot Learning Benchmark and Learning Environment*. arXiv: 1909.12271 [cs.RO] (cit. on p. 54).

Johnson, Matthew, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta (2016). "Composing graphical models with neural networks for structured representations and fast inference." In: *Neural Information Processing Systems (NIPS)* (cit. on pp. 25, 28).

Kaelbling, Leslie Pack. "Learning to achieve goals." In: Citeseer (cit. on p. 173).

Kakade, Sham Machandranath et al. (2003). "On the sample complexity of reinforcement learning." Doctoral dissertation (cit. on p. 38).

Kalashnikov, Dmitry, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. (2018). "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation." In: *arXiv preprint arXiv:1806.10293* (cit. on p. 54).

Kearns, Michael and Satinder Singh (2002). "Near-optimal reinforcement learning in polynomial time." In: *Machine learning* 49.2-3, pp. 209–232 (cit. on p. 38).

Khetarpal, Khimya, Matthew Riemer, Irina Rish, and Doina Precup (2020). "Towards continual reinforcement learning: A review and perspectives." In: *arXiv preprint arXiv:2012.13490* (cit. on p. 85).

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization." In: *arXiv preprint arXiv:1412.6980* (cit. on pp. 128, 131).

Kingma, Diederik P and Max Welling (2013). "Auto-encoding variational bayes." In: *arXiv preprint arXiv:1312.6114* (cit. on pp. 25, 28, 31).

Kirkpatrick, James, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. (2017). “Overcoming catastrophic forgetting in neural networks.” In: *Proceedings of the national academy of sciences* 114.13, pp. 3521–3526 (cit. on p. 85).

Klyubin, Alexander S, Daniel Polani, and Chrystopher L Nehaniv (2005). “Empowerment: A universal agent-centric measure of control.” In: *2005 IEEE Congress on Evolutionary Computation*. Vol. 1. IEEE, pp. 128–135 (cit. on p. 40).

Kober, Jens and Jan Peters (2008). “Policy search for motor primitives in robotics.” In: *Advances in neural information processing systems* 21 (cit. on p. 71).

Kober, Jens and Jan Peters (2012). “Reinforcement learning in robotics: A survey.” In: *Reinforcement Learning*. Springer, pp. 579–610 (cit. on p. 5).

Koch, Gregory (2015). “Siamese neural networks for one-shot image recognition.” In: (cit. on p. 8).

Konda, Vijay R. and John N. Tsitsiklis (1999). “Actor-Critic Algorithms.” In: *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*. Ed. by Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller. The MIT Press, pp. 1008–1014. URL: `http://papers.nips.cc/paper/1786-actor-critic-algorithms` (cit. on p. 55).

Kostrikov, Ilya, Denis Yarats, and Rob Fergus (2020). “Image augmentation is all you need: Regularizing deep reinforcement learning from pixels.” In: *arXiv preprint arXiv:2004.13649* (cit. on pp. 77, 163).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “Imagenet classification with deep convolutional neural networks.” In: *Advances in neural information processing systems* 25 (cit. on p. 2).

Kuefler, Alex and Mykel J. Kochenderfer (2018). “Burn-In Demonstrations for Multi-Modal Imitation Learning.” In: (cit. on p. 7).

Kumar, Vikash, Abhishek Gupta, Emanuel Todorov, and Sergey Levine (2016a). “Learning dexterous manipulation policies from experience and imitation.” In: *arXiv preprint arXiv:1611.05095* (cit. on p. 54).

Kumar, Vikash, Yuval Tassa, Tom Erez, and Emanuel Todorov (2014a). “Real-time behaviour synthesis for dynamic hand-manipulation.” In: *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, pp. 6808–6815. DOI: `10.1109/ICRA.2014.6907864`. URL: `https://doi.org/10.1109/ICRA.2014.6907864` (cit. on p. 54).

Kumar, Vikash, Yuval Tassa, Tom Erez, and Emanuel Todorov (2014b). “Real-time behaviour synthesis for dynamic hand-manipulation.” In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 6808–6815 (cit. on p. 71).

Kumar, Vikash, Emanuel Todorov, and Sergey Levine (2016b). "Optimal control with learned local models: Application to dexterous manipulation." In: *2016 IEEE International Conference on Robotics and Automation, ICRA 2016, Stockholm, Sweden, May 16-21, 2016*. Ed. by Danica Kragic, Antonio Bicchi, and Alessandro De Luca. IEEE, pp. 378–383. DOI: 10.1109/ICRA.2016.7487156. URL: https://doi.org/10.1109/ICRA.2016.7487156 (cit. on pp. 52, 54, 66).

Lacoste, Alexandre, Thomas Boquet, Negar Rostamzadeh, Boris Oreshki, Wonchang Chung, and David Krueger (2017). "Deep Prior." In: *arXiv preprint arXiv:1712.05016* (cit. on p. 25).

Lake, Brenden M, Ruslan Salakhutdinov, and Joshua B Tenenbaum (2015). "Human-level concept learning through probabilistic program induction." In: *Science* 350.6266, pp. 1332–1338 (cit. on p. 24).

Laskin, Michael, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas (2020). "Reinforcement learning with augmented data." In: *arXiv preprint arXiv:2004.14990* (cit. on p. 163).

Lawrence, Neil D and John C Platt (2004). "Learning to learn with the informative vector machine." In: *International Conference on Machine Learning (ICML)*, p. 65 (cit. on p. 24).

Lee, Lisa, Benjamin Eysenbach, Emilio Parisotto, Eric P. Xing, Sergey Levine, and Ruslan Salakhutdinov (2019a). "Efficient Exploration via State Marginal Matching." In: *CoRR* abs/1906.05274. arXiv: 1906.05274. URL: http://arxiv.org/abs/1906.05274 (cit. on p. 58).

Lee, Lisa, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov (2019b). "Efficient exploration via state marginal matching." In: *arXiv preprint arXiv:1906.05274* (cit. on p. 41).

Levine, Sergey, Chelsea Finn, Trevor Darrell, and Pieter Abbeel (2016a). "End-to-end training of deep visuomotor policies." In: *The Journal of Machine Learning Research* 17.1, pp. 1334–1373 (cit. on pp. 2, 5, 42, 83).

Levine, Sergey, Aviral Kumar, George Tucker, and Justin Fu (2020). "Offline reinforcement learning: Tutorial, review, and perspectives on open problems." In: *arXiv preprint arXiv:2005.01643* (cit. on p. 100).

Levine, Sergey, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen (2016b). "Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection." In: *CoRR* abs/1603.02199. arXiv: 1603.02199. URL: http://arxiv.org/abs/1603.02199 (cit. on p. 54).

Levine, Sergey, Zoran Popovic, and Vladlen Koltun (2011). "Nonlinear inverse reinforcement learning with gaussian processes." In: *Advances in Neural Information Processing Systems*, pp. 19–27 (cit. on p. 18).

Li, Ke and Jitendra Malik (2017). "Learning to optimize neural nets." In: *arXiv preprint arXiv:1703.00441* (cit. on p. 8).

Li, Kevin, Abhishek Gupta, Ashwin Reddy, Vitchyr H Pong, Aurick Zhou, Justin Yu, and Sergey Levine (2021). "MURAL: Meta-Learning Uncertainty-Aware Rewards for Outcome-Driven Reinforcement Learning." In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 6346–6356. URL: https://proceedings.mlr.press/v139/li21g.html (cit. on p. 77).

Li, Kun and Joel W. Burdick (2017). "Meta Inverse Reinforcement Learning via Maximum Reward Sharing for Human Motion Analysis." In: *CoRR* abs/1710.03592 (cit. on p. 8).

Li, Yunzhu, Jiaming Song, and Stefano Ermon (2017a). "Inferring the latent structure of human decision-making from raw visual inputs." In: *arXiv preprint arXiv:1703.08840* (cit. on p. 7).

Li, Zhenguo, Fengwei Zhou, Fei Chen, and Hang Li (2017b). "Meta-SGD: Learning to Learn Quickly for Few Shot Learning." In: *arXiv preprint arXiv:1707.09835* (cit. on p. 140).

Lillicrap, Timothy P, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra (2015). "Continuous control with deep reinforcement learning." In: *arXiv preprint arXiv:1509.02971* (cit. on p. 88).

Losey, Dylan P and Marcia K O'Malley (2018). "Including uncertainty when learning from human corrections." In: *Conference on Robot Learning*. PMLR, pp. 123–132 (cit. on p. 72).

Lowrey, Kendall, Svetoslav Kolev, Jeremy Dao, Aravind Rajeswaran, and Emanuel Todorov (2018). "Reinforcement learning for non-prehensile manipulation: Transfer from simulation to physical system." In: *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*. IEEE, pp. 35–42 (cit. on p. 71).

Lu, Kevin, Aditya Grover, Pieter Abbeel, and Igor Mordatch (2020). "Reset-free lifelong learning with skill-space planning." In: *arXiv preprint arXiv:2012.03548* (cit. on p. 85).

MacKay, David JC (1992). "A practical Bayesian framework for backpropagation networks." In: *Neural computation* (cit. on p. 25).

Mandikal, Priyanka and Kristen Grauman (2020a). *Dexterous Robotic Grasping with Object-Centric Visual Affordances*. arXiv: 2009.01439 [cs.RO] (cit. on p. 54).

Mandikal, Priyanka and Kristen Grauman (2020b). “Learning Dexterous Grasping with Object-Centric Visual Affordances.” In: *arXiv preprint arXiv:2009.01439* (cit. on p. 71).

Mendez, Jorge, Boyu Wang, and Eric Eaton (2020). “Lifelong Policy Gradient Learning of Factored Policies for Faster Training Without Forgetting.” In: *Advances in Neural Information Processing Systems* 33 (cit. on p. 85).

Mishra, Nikhil, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel (2017). “Meta-learning with temporal convolutions.” In: *arXiv preprint arXiv:1707.03141* (cit. on pp. 8, 10).

Mishra, Nikhil, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel (2018). “A Simple Neural Attentive Meta-Learner.” In: *International Conference on Learning Representations* (cit. on pp. 25, 140).

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller (2013). “Playing atari with deep reinforcement learning.” In: *arXiv preprint arXiv:1312.5602* (cit. on p. 42).

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. (2015). “Human-level control through deep reinforcement learning.” In: *nature* 518.7540, pp. 529–533 (cit. on pp. 5, 88).

Mohamed, Shakir and Danilo Jimenez Rezende (2015). “Variational information maximisation for intrinsically motivated reinforcement learning.” In: *Advances in neural information processing systems*, pp. 2125–2133 (cit. on p. 40).

Moldovan, Teodor Mihai and Pieter Abbeel (2012). “Safe exploration in markov decision processes.” In: *arXiv preprint arXiv:1205.4810* (cit. on pp. 90, 174).

Montgomery, William, Anurag Ajay, Chelsea Finn, Pieter Abbeel, and Sergey Levine (2016). “Reset-Free Guided Policy Search: Efficient Deep Reinforcement Learning with Stochastic Initial States.” In: *CoRR* abs/1610.01112. arXiv: `1610.01112`. URL: `http://arxiv.org/abs/1610.01112` (cit. on p. 54).

Mordatch, Igor (2018). “Concept Learning with Energy-Based Models.” In: *ICLR Workshop* (cit. on p. 8).

Mordatch, Igor, Zoran Popović, and Emanuel Todorov (2012). “Contact-invariant optimization for hand manipulation.” In: *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, pp. 137–144 (cit. on pp. 54, 71).

Myers, Vivek, Erdem Biyik, Nima Anari, and Dorsa Sadigh (2022). “Learning multimodal rewards from rankings.” In: *Conference on Robot Learning*. PMLR, pp. 342–352 (cit. on p. 72).

Nachum, Ofir, Shixiang Shane Gu, Honglak Lee, and Sergey Levine (2018). "Data-efficient hierarchical reinforcement learning." In: *Advances in Neural Information Processing Systems*, pp. 3303–3313 (cit. on p. 41).

Nagabandi, Anusha, Kurt Konolige, Sergey Levine, and Vikash Kumar (2020). "Deep dynamics models for learning dexterous manipulation." In: *Conference on Robot Learning*, pp. 1101–1112 (cit. on pp. 52, 54, 62, 66, 74).

Naik, Devang K and RJ Mammone (1992). "Meta-neural networks that learn by learning." In: *Neural Networks, 1992. IJCNN., International Joint Conference on*. Vol. 1. IEEE, pp. 437–442 (cit. on p. 8).

Nair, Ashvin, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine (2018). "Visual Reinforcement Learning with Imagined Goals." In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (cit. on p. 54).

Neal, Radford M (1995). *Bayesian learning for neural networks*. PhD thesis, University of Toronto (cit. on p. 25).

Nemec, Bojan, Leon Zlajpah, and Ales Ude (2017). "Door opening by joining reinforcement learning and intelligent control." In: *18th International Conference on Advanced Robotics, ICAR 2017, Hong Kong, China, July 10-12, 2017*. IEEE, pp. 222–228. DOI: 10.1109/ICAR.2017.8023522. URL: https://doi.org/10.1109/ICAR.2017.8023522 (cit. on p. 54).

Ng, Andrew Y, Daishi Harada, and Stuart Russell (1999). "Policy invariance under reward transformations: Theory and application to reward shaping." In: (cit. on p. 5).

Ng, Andrew Y. and Stuart J. Russell (2000). "Algorithms for Inverse Reinforcement Learning." In: *Proceedings of the Seventeenth International Conference on Machine Learning*. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. (cit. on pp. 5, 6).

Nichol, Alex and John Schulman (2018). "Reptile: a Scalable Metalearning Algorithm." In: *arXiv preprint arXiv:1803.02999* (cit. on p. 140).

Okamura, Allison M, Niels Smaby, and Mark R Cutkosky (2000). "An overview of dexterous manipulation." In: *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*. Vol. 1. IEEE, pp. 255–262 (cit. on p. 54).

OpenAI (2018). "Learning Dexterous In-Hand Manipulation." In: *CoRR* abs/1808.00177. arXiv: 1808.00177. URL: http://arxiv.org/abs/1808.00177 (cit. on p. 54).

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 (2018). "Learning Dexterous In-Hand Manipulation." In: *CoRR* abs/1808.00177. arXiv: 1808.00177. URL: http://arxiv.org/abs/1808.00177 (cit. on pp. 54, 71).

Parisi, German I, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter (2019). "Continual lifelong learning with neural networks: A review." In: *Neural Networks* 113, pp. 54–71 (cit. on p. 85).

Parisotto, Emilio, Lei Jimmy Ba, and Ruslan Salakhutdinov (2016). "Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning." In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1511.06342 (cit. on pp. 52, 55).

Parr, Ronald and Stuart J Russell (1998). "Reinforcement learning with hierarchies of machines." In: *Advances in neural information processing systems*, pp. 1043–1049 (cit. on p. 41).

Pathak, Deepak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell (2017). "Curiosity-driven exploration by self-supervised prediction." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 16–17 (cit. on p. 40).

Peng, Xue Bin, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine (2020). "Learning agile robotic locomotion skills by imitating animals." In: *arXiv preprint arXiv:2004.00784* (cit. on p. 54).

Platanios, Emmanouil Antonios, Abulhair Saparov, and Tom Mitchell (2020). "Jelly bean world: A testbed for never-ending learning." In: *arXiv preprint arXiv:2002.06306* (cit. on p. 85).

Ploeger, Kai, Michael Lutter, and Jan Peters (2020). *High Acceleration Reinforcement Learning for Real-World Juggling with Binary Rewards*. arXiv: 2010.13483 [cs.RO] (cit. on pp. 52, 54).

Pong, Vitchyr H, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine (2019). "Skew-fit: State-covering self-supervised reinforcement learning." In: *arXiv preprint arXiv:1903.03698* (cit. on pp. 42, 85).

Posa, Michael, Cecilia Cantu, and Russ Tedrake (2014). "A direct method for trajectory optimization of rigid bodies through contact." In: *The International Journal of Robotics Research* 33.1, pp. 69–81 (cit. on p. 71).

Precup, Doina (2001). "Temporal abstraction in reinforcement learning." In: (cit. on p. 41).

Racaniere, Sebastien, Andrew K Lampinen, Adam Santoro, David P Reichert, Vlad Firoiu, and Timothy P Lillicrap (2019). "Automated curricula through setter-solver interactions." In: *arXiv preprint arXiv:1909.12892* (cit. on p. 41).

Rajeswaran, Aravind, Vikash Kumar, Abhishek Gupta, John Schulman, Emanuel Todorov, and Sergey Levine (2017a). "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations." In: *CoRR* abs/1709.10087. arXiv: 1709.10087. URL: http://arxiv.org/abs/1709.10087 (cit. on p. 54).

Rajeswaran, Aravind, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine (2017b). "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations." In: *arXiv preprint arXiv:1709.10087* (cit. on pp. 42, 71).

Ramachandran, Deepak and Eyal Amir (2007). "Bayesian Inverse Reinforcement Learning." In: *International Joint Conference on Artifical Intelligence*. San Francisco, CA, USA (cit. on pp. 6, 7).

Ratliff, Nathan D, J Andrew Bagnell, and Martin A Zinkevich (2006a). "Maximum margin planning." In: *Proceedings of the 23rd international conference on Machine learning*. ACM, pp. 729–736 (cit. on p. 6).

Ratliff, Nathan D., J. Andrew Bagnell, and Martin Zinkevich (2006b). "Maximum margin planning." In: *Machine Learning, Proceedings of the Twenty-Third International Conference ICML*. DOI: 10.1145/1143844.1143936 (cit. on p. 72).

Ravi, Sachin and Hugo Larochelle (2016). "Optimization as a model for few-shot learning." In: (cit. on pp. 8, 10).

Ravi, Sachin and Hugo Larochelle (2017). "Optimization as a model for few-shot learning." In: *International Conference on Learning Representations (ICLR)* (cit. on pp. 3, 25, 139, 140).

Reddy, Siddharth, Anca D Dragan, and Sergey Levine (2019). "Sqil: Imitation learning via reinforcement learning with sparse rewards." In: *arXiv preprint arXiv:1905.11108* (cit. on p. 72).

Reed, Scott, Yutian Chen, Thomas Paine, Aaron van den Oord, Oriol Vinyals, SM Ali Eslami, Danilo Rezende, and Nando de Freitas (2018). "Few-shot Autoregressive Density Estimation: Towards Learning to Learn Distributions." In: *ICLR* (cit. on p. 8).

Rezende, Danilo Jimenez, Shakir Mohamed, Ivo Danihelka, Karol Gregor, and Daan Wierstra (2016). "One-shot generalization in deep generative models." In: *arXiv preprint arXiv:1603.05106* (cit. on p. 8).

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). “Stochastic backpropagation and approximate inference in deep generative models.” In: *arXiv preprint arXiv:1401.4082* (cit. on p. 25).

Richter, Charles and Nicholas Roy (2017). “Safe Visual Navigation via Deep Learning and Novelty Detection.” In: *Robotics: Science and Systems XIII, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, July 12-16, 2017*. Ed. by Nancy M. Amato, Siddhartha S. Srinivasa, Nora Ayanian, and Scott Kuindersma. DOI: 10.15607/RSS.2017.XIII.064. URL: http://www.roboticsproceedings.org/rss13/p64.html (cit. on p. 52).

Riedmiller, Martin A., Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Van de Wiele, Volodymyr Mnih, Nicolas Heess, and Jost Tobias Springenberg (2018). “Learning by Playing - Solving Sparse Reward Tasks from Scratch.” In: *CoRR* abs/1802.10567. arXiv: 1802.10567. URL: http://arxiv.org/abs/1802.10567 (cit. on p. 55).

Ross, Stéphane, Geoffrey Gordon, and Drew Bagnell (2011). “A reduction of imitation learning and structured prediction to no-regret online learning.” In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635 (cit. on p. 133).

Ross, Stéphane, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J. Andrew Bagnell, and Martial Hebert (2013). “Learning monocular reactive UAV control in cluttered natural environments.” In: *2013 IEEE International Conference on Robotics and Automation*. DOI: 10.1109/ICRA.2013.6630809 (cit. on p. 72).

Ruder, Sebastian (2017). “An Overview of Multi-Task Learning in Deep Neural Networks.” In: *CoRR* abs/1706.05098. arXiv: 1706.05098. URL: http://arxiv.org/abs/1706.05098 (cit. on p. 55).

Rusu, Andrei A., Sergio Gomez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell (2016a). “Policy Distillation.” In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1511.06295 (cit. on pp. 52, 55).

Rusu, Andrei A, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell (2016b). “Progressive neural networks.” In: *arXiv preprint arXiv:1606.04671* (cit. on p. 85).

Salge, Christoph, Cornelius Glackin, and Daniel Polani (2014). "Empowerment–an introduction." In: *Guided Self-Organization: Inception*. Springer, pp. 67–114 (cit. on p. 40).

Samvelyan, Mikayel, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson (2019). "The starcraft multi-agent challenge." In: *arXiv preprint arXiv:1902.04043* (cit. on p. 85).

Santoro, Adam, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap (2016). "Meta-learning with memory-augmented neural networks." In: *International conference on machine learning* (cit. on pp. 8, 15, 25).

Santos, Reginaldo J. (1996b). "Equivalence of regularization and truncated iteration for general ill-posed problems." In: *Linear Algebra and its Applications* (cit. on p. 29).

Santos, Reginaldo J (1996a). "Equivalence of regularization and truncated iteration for general ill-posed problems." In: *Linear algebra and its applications* 236, pp. 25–33 (cit. on p. 13).

Schaul, Tom, Daniel Horgan, Karol Gregor, and David Silver (2015). "Universal value function approximators." In: *International Conference on Machine Learning*, pp. 1312–1320 (cit. on p. 173).

Schmidhuber, Jurgen (1987a). "Evolutionary Principles in Self-Referential Learning. On Learning now to Learn: The Meta-Meta-Meta...-Hook." Diploma Thesis. Technische Universitat Munchen, Germany. URL: `http://www.idsia.ch/~juergen/diploma.html` (cit. on p. 85).

Schmidhuber, Jürgen (1987b). "Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook." Doctoral dissertation. Technische Universität München (cit. on pp. 3, 8, 24).

Schmidhuber, Jürgen (1991). "Curious model-building control systems." In: *Proc. international joint conference on neural networks*, pp. 1458–1463 (cit. on p. 40).

Schwarz, Jonathan, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell (2018). "Progress & compress: A scalable framework for continual learning." In: *International Conference on Machine Learning*. PMLR, pp. 4528–4537 (cit. on p. 85).

Sener, Ozan and Vladlen Koltun (2018). "Multi-Task Learning as Multi-Objective Optimization." In: *CoRR* abs/1810.04650. arXiv: `1810.04650`. URL: `http://arxiv.org/abs/1810.04650` (cit. on pp. 52, 55).

Sharma, Archit, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman (2019). "Dynamics-aware unsupervised discovery of skills." In: *arXiv preprint arXiv:1907.01657* (cit. on pp. 38, 40, 42, 48, 49, 51, 85, 148, 156).

Sharma, Archit, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn (2021a). “Persistent Reinforcement Learning via Subgoal Curricula.” In: *arXiv preprint arXiv:2107.12931* (cit. on pp. 72, 74, 84, 86, 92, 95, 173, 175).

Sharma, Archit, Kelvin Xu, Nikhil Sardana, Abhishek Gupta, Karol Hausman, Sergey Levine, and Chelsea Finn (2021b). “Autonomous Reinforcement Learning: Formalism and Benchmarking.” In: *arXiv preprint arXiv:2112.09605* (cit. on p. 72).

Shelhamer, E., J. Long, and T. Darrell (2017). “Fully Convolutional Networks for Semantic Segmentation.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.4 (cit. on p. 7).

Shu, Rui, Hung H Bui, Shengjia Zhao, Mykel J Kochenderfer, and Stefano Ermon (2018). “Amortized Inference Regularization.” In: *arXiv preprint arXiv:1805.08913* (cit. on pp. 25, 28).

Shyam, Pranav, Shubham Gupta, and Ambedkar Dukkipati (2017). “Attentive recurrent comparators.” In: *arXiv preprint arXiv:1703.00767* (cit. on p. 8).

Siegelmann, Hava T and Eduardo D Sontag (1995). “On the computational power of neural nets.” In: *Journal of computer and system sciences* 50.1, pp. 132–150 (cit. on p. 10).

Silver, David, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. (2016). “Mastering the game of Go with deep neural networks and tree search.” In: *nature* 529.7587, pp. 484–489 (cit. on pp. 5, 85).

Silver, David, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. (2018). “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play.” In: *Science* 362.6419, pp. 1140–1144 (cit. on p. 41).

Singh, Avi, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine (2019). “End-to-end robotic reinforcement learning without reward engineering.” In: *arXiv preprint arXiv:1904.07854* (cit. on p. 80).

Sjöberg, Jonas and Lennart Ljung (1995). “Overtraining, regularization and searching for a minimum, with application to neural networks.” In: *International Journal of Control* 62.6, pp. 1391–1407 (cit. on p. 14).

Smith, Laura, Nikita Dhawan, Marvin Zhang, Pieter Abbeel, and Sergey Levine (2019). “Avid: Learning multi-stage tasks via pixel-level translation of human videos.” In: *arXiv preprint arXiv:1912.04443* (cit. on pp. 54, 58).

Snell, Jake, Kevin Swersky, and Richard Zemel (2017). “Prototypical networks for few-shot learning.” In: *Advances in Neural Information Processing Systems*, pp. 4080–4090 (cit. on pp. 8, 140).

Song, Shuran, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser (2017). “Semantic Scene Completion from a Single Depth Image.” In: *IEEE Conference on Computer Vision and Pattern Recognition* (cit. on p. 16).

Srinivasan, Krishnan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn (2020). “Learning to be safe: Deep rl with a safety critic.” In: *arXiv preprint arXiv:2010.14603* (cit. on p. 75).

Sukhbaatar, Sainbayar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus (2017a). “Intrinsic motivation and automatic curricula via asymmetric self-play.” In: *arXiv preprint arXiv:1703.05407* (cit. on pp. 41, 50).

Sukhbaatar, Sainbayar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus (2017b). “Intrinsic motivation and automatic curricula via asymmetric self-play.” In: *arXiv preprint arXiv:1703.05407* (cit. on p. 52).

Sung, Flood, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales (2017). “Learning to Compare: Relation Network for Few-Shot Learning.” In: *CoRR* abs/1711.06025. arXiv: `1711.06025`. URL: `http://arxiv.org/abs/1711.06025` (cit. on p. 140).

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014). “Sequence to sequence learning with neural networks.” In: *Advances in neural information processing systems* 27 (cit. on p. 2).

Sutton, Richard S and Andrew G Barto (2018a). *Reinforcement learning: An introduction* (cit. on p. 55).

Sutton, Richard S and Andrew G Barto (2018b). *Reinforcement learning: An introduction*. MIT press (cit. on pp. 86, 87).

Sutton, Richard S, Doina Precup, and Satinder Singh (1999). “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning.” In: *Artificial intelligence* 112.1-2, pp. 181–211 (cit. on pp. 38, 41).

Synnaeve, Gabriel, Nantas Nardelli, Alex Auvolat, Soumith Chintala, Timothée Lacroix, Zeming Lin, Florian Richoux, and Nicolas Usunier (2016). “Torchcraft: a library for machine learning research on real-time strategy games.” In: *arXiv preprint arXiv:1611.00625* (cit. on p. 129).

Tang, Haoran, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel (2017). “# Exploration: A study of count-based exploration for deep reinforcement learning.” In: *Advances in neural information processing systems*, pp. 2753–2762 (cit. on p. 40).

Tassa, Yuval, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. (2018). "Deepmind control suite." In: *arXiv preprint arXiv:1801.00690* (cit. on p. 94).

Teh, Yee Whye, Victor Bapst, Wojciech M. Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu (2017). "Distral: Robust multitask reinforcement learning." In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, pp. 4496–4506. URL: http://papers.nips.cc/paper/7036-distral-robust-multitask-reinforcement-learning (cit. on pp. 52, 55).

Tenenbaum, Joshua Brett (1999). "A Bayesian framework for concept learning." Doctoral dissertation. Massachusetts Institute of Technology (cit. on p. 24).

Tesauro, Gerald (1995). "Temporal difference learning and TD-Gammon." In: (cit. on p. 41).

Thrun, Sebastian and Tom M Mitchell (1995). "Lifelong robot learning." In: *Robotics and autonomous systems* 15.1-2, pp. 25–46 (cit. on p. 85).

Thrun, Sebastian and Lorien Pratt (1998). *Learning to learn*. Kluwer Academic Publishers (cit. on p. 3).

Thrun, Sebastian and Lorien Pratt (2012). *Learning to learn*. Springer Science & Business Media (cit. on p. 8).

Todorov, Emanuel, Tom Erez, and Yuval Tassa (2012). "Mujoco: A physics engine for model-based control." In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 5026–5033 (cit. on p. 85).

Urakami, Yusuke, Alec Hodgkinson, Casey Carlin, Randall Leu, Luca Rigazio, and Pieter Abbeel (2019). "DoorGym: A Scalable Door Opening Environment And Baseline Agent." In: *CoRR* abs/1908.01887. arXiv: 1908.01887. URL: http://arxiv.org/abs/1908.01887 (cit. on p. 54).

Van Hasselt, Hado, Arthur Guez, and David Silver (2016). "Deep reinforcement learning with double q-learning." In: *Thirtieth AAAI conference on artificial intelligence* (cit. on pp. 142, 146, 150, 154).

Van Hoof, Herke, Tucker Hermans, Gerhard Neumann, and Jan Peters (2015). "Learning robot in-hand manipulation with tactile features." In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 121–127 (cit. on p. 71).

Vecerik, Mel, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller (2017). "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards.” In: *arXiv preprint arXiv:1707.08817* (cit. on p. 42).

Vezhnevets, Alexander Sasha, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu (2017). “Feudal networks for hierarchical reinforcement learning.” In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, pp. 3540–3549 (cit. on p. 41).

Villani, Valeria, Fabio Pini, Francesco Leali, and Cristian Secchi (2018). “Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications.” In: *Mechatronics* 55, pp. 248–266 (cit. on p. 72).

Vinyals, Oriol, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. (2016). “Matching networks for one shot learning.” In: *Advances in Neural Information Processing Systems*, pp. 3630–3638 (cit. on pp. 8, 25, 139, 140).

Von Stackelberg, Heinrich (1934). *Marktform und Gleichgewicht* (cit. on p. 44).

Wan, Jing, Zhilin Zhang, Jingwen Yan, Taiyong Li, Bhaskar D Rao, Shiaofen Fang, Sungeun Kim, Shannon L Risacher, Andrew J Saykin, and Li Shen (2012). “Sparse Bayesian multi-task learning for predicting cognitive outcomes from neuroimaging measures in Alzheimer’s disease.” In: *Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 24).

Wang, Jane X, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick (2016). “Learning to reinforcement learn.” In: *arXiv preprint arXiv:1611.05763* (cit. on pp. 8, 25).

Wang, Rose E, Sarah A Wu, James A Evans, Joshua B Tenenbaum, David C Parkes, and Max Kleiman-Weiner (2020). “Too many cooks: Coordinating multi-agent collaboration through inverse planning.” In: *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2032–2034 (cit. on p. 85).

Wang, Rui, Joel Lehman, Jeff Clune, and Kenneth O Stanley (2019). “Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions.” In: *arXiv preprint arXiv:1901.01753* (cit. on p. 41).

Wang, Yuxiong and Martial Hebert (2016). “Learning to Learn: Model Regression Networks for Easy Small Sample Learning.” In: *European Conference on Computer Vision (ECCV)* (cit. on p. 8).

Wiering, Marco and Jürgen Schmidhuber (1997). “HQ-learning.” In: *Adaptive Behavior* 6.2, pp. 219–246 (cit. on p. 41).

Wołczyk, Maciej, Michał Zajc, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś (2021). “Continual World: A Robotic Benchmark For Continual Reinforcement Learning.” In: *arXiv preprint arXiv:2105.10919* (cit. on pp. 85, 87).

Wu, Bohan, Iretiayo Akinola, Jacob Varley, and Peter Allen (2019). *MAT: Multi-Fingered Adaptive Tactile Grasping via Deep Reinforcement Learning*. arXiv: 1909.04787 [cs.RO] (cit. on p. 54).

Wulfmeier, M., D. Z. Wang, and I. Posner (2016a). "Watch this: Scalable cost-function learning for path planning in urban environments." In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2089–2095 (cit. on p. 7).

Wulfmeier, Markus, Abbas Abdolmaleki, Roland Hafner, Jost Tobias Springenberg, Michael Neunert, Tim Hertweck, Thomas Lampe, Noah Y. Siegel, Nicolas Heess, and Martin A. Riedmiller (2019). "Regularized Hierarchical Policies for Compositional Transfer in Robotics." In: *CoRR* abs/1906.11228. arXiv: 1906.11228. URL: http://arxiv.org/abs/1906.11228 (cit. on p. 55).

Wulfmeier, Markus, Peter Ondruska, and Ingmar Posner (2015a). "Maximum Entropy Deep Inverse Reinforcement Learning." In: *Neural Information Processing Systems Conference, Deep Reinforcement Learning Workshop*. Vol. abs/1507.04888 (cit. on pp. 6, 18).

Wulfmeier, Markus, Peter Ondruska, and Ingmar Posner (2015b). "Maximum entropy deep inverse reinforcement learning." In: *arXiv preprint arXiv:1507.04888* (cit. on p. 72).

Wulfmeier, Markus, Dushyant Rao, and Ingmar Posner (2016b). "Incorporating Human Domain Knowledge into Large Scale Cost Function Learning." In: *CoRR* abs/1612.04318 (cit. on p. 7).

Xie, Annie, James Harrison, and Chelsea Finn (2020). "Deep reinforcement learning amidst lifelong non-stationarity." In: *arXiv preprint arXiv:2006.10701* (cit. on pp. 83, 85).

Xu, Kelvin, Siddharth Verma, Chelsea Finn, and Sergey Levine (2020). "Continual Learning of Control Primitives: Skill Discovery via Reset-Games." In: *arXiv preprint arXiv:2011.05286* (cit. on pp. 72, 83, 85).

Xu, Zhe, Vikash Kumar, and Emanuel Todorov (2013). "A low-cost and modular, 20-DOF anthropomorphic robotic hand: Design, actuation and modeling." In: *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 368–375 (cit. on p. 71).

Yahya, Ali, Adrian Li, Mrinal Kalakrishnan, Yevgen Chebotar, and Sergey Levine (2017). "Collective robot reinforcement learning with distributed asynchronous guided policy search." In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 79–86 (cit. on pp. 2, 42, 83).

Yamane, Katsu, James J Kuffner, and Jessica K Hodgins (2004). "Synthesizing animations of human manipulation tasks." In: *ACM SIGGRAPH 2004 Papers*, pp. 532–539 (cit. on p. 54).

Yang, Ruihan, Huazhe Xu, Yi Wu, and Xiaolong Wang (2020). “Multi-Task Reinforcement Learning with Soft Modularization.” In: *CoRR* abs/2003.13661. arXiv: 2003.13661. URL: https://arxiv.org/abs/2003.13661 (cit. on p. 55).

Yu, Kai, Volker Tresp, and Anton Schwaighofer (2005). “Learning Gaussian processes from multiple tasks.” In: *International Conference on Machine Learning (ICML)* (cit. on p. 24).

Yu, Tianhe, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn (2020a). “Gradient Surgery for Multi-Task Learning.” In: *CoRR* abs/2001.06782. arXiv: 2001.06782. URL: https://arxiv.org/abs/2001.06782 (cit. on pp. 52, 55).

Yu, Tianhe, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine (2019). “Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning.” In: *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*. Ed. by Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura. Vol. 100. Proceedings of Machine Learning Research. PMLR, pp. 1094–1100. URL: http://proceedings.mlr.press/v100/yu20a.html (cit. on pp. 52, 55).

Yu, Tianhe, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine (2020b). “Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning.” In: *Conference on Robot Learning*. PMLR, pp. 1094–1100 (cit. on pp. 84, 85, 92).

Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals (2014). “Recurrent neural network regularization.” In: *arXiv preprint arXiv:1409.2329* (cit. on p. 128).

Zeng, Andy, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser (2020). “Tossingbot: Learning to throw arbitrary objects with residual physics.” In: *IEEE Transactions on Robotics* 36.4, pp. 1307–1319 (cit. on pp. 2, 83).

Zhu, Henry, Abhishek Gupta, Aravind Rajeswaran, Sergey Levine, and Vikash Kumar (2019). “Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost.” In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3651–3657 (cit. on pp. 52, 54, 66, 71, 74, 142, 150).

Zhu, Henry, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey Levine (2020a). “The Ingredients of Real World Robotic Reinforcement Learning.” In: *International Conference on Learning Representations* (cit. on pp. 40, 47–49, 83–86, 95, 145, 153, 175).

Zhu, Henry, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey Levine (2020b). “The Ingredients of Real-World Robotic

Reinforcement Learning." In: *arXiv preprint arXiv:2004.12570* (cit. on pp. 52, 54, 58, 65, 66, 73, 81).

Ziebart, Brian D., Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey (2008a). "Maximum Entropy Inverse Reinforcement Learning." In: *AAAI*. AAAI Press. ISBN: 978-1-57735-368-3 (cit. on p. 72).

Ziebart, Brian D., Andrew Maas, J. Andrew Bagnell, and Anind K. Dey (2008b). "Maximum Entropy Inverse Reinforcement Learning." In: *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*. AAAI'08. Chicago, Illinois: AAAI Press (cit. on pp. 6, 7, 9, 11, 12, 21).

Zolna, Konrad, Alexander Novikov, Ksenia Konyushkova, Caglar Gulcehre, Ziyu Wang, Yusuf Aytar, Misha Denil, Nando de Freitas, and Scott Reed (2020). "Offline learning from demonstrations and unlabeled experience." In: *arXiv preprint arXiv:2011.13885* (cit. on p. 72).

A

APPENDIX: LEARNING A PRIOR OVER INTENT VIA META-INVERSE REINFORCEMENT LEARNING

A.1 SPRITEWORLD EXPERIMENTAL DETAILS

We first describe the experimental details here specific to the SpriteWorld domain experiments.

A.1.1 *Algoritmic Details*

The input to our reward function for all experiments in this domain is a 80×80 RGB image, with an output space of 400 in the underlying MDP state space. We parameterize the reward function for all methods starting from the same base learner whose architecture we summarize in Table 6.

Our LSTM (Hochreiter and Schmidhuber, 1997) implementation is based on the variant used in Zaremba et al. (2014). The input to the LSTM at each time step is the location of the agent, embedded as the (x, y)-coordinates. This is used to predict an spatial map fed as input to the base CNN. We also experimented with conditioning the initial hidden state on image features from a separate CNN, but found that this did not improve performance.

In our demo conditional model, we preserve the spatial information of the demonstrations by feeding in the state visitation map as a image-grid, upsampled with bilinear interpolation, as an additional channel to the image. In our setup, both the demo-conditional models share the same convolutional architecture, but differ only in how they encode condition on the demonstrations.

For all our methods, we optimized our model with Adam (Kingma and Ba, 2014). We tuned over the learning rate α, the inner learning rate β and ℓ_2 weight decay on the initial parameters. We initialize our models with the Glorot initialization Glorot and Bengio, 2010. In our LSTM learner, we tuned over embedding sizes and dimensionality.

A negative result we found was that bias transformation (Finn et al., 2017d) did not help in our experimental setting.

Table 6: Hyperparameter summary on Spriteworld environment. Curly brackets indicate the parameter was chosen from that set.

Hyperparameters	Value
Architecture	Conv($256-8\times8-2$)
	Conv($128-4\times4-2$)
	Conv($64-3\times3-1$)
	Conv($64-3\times3-1$)
	Conv($1-1\times1-1$)
Learning rate α	{0.0001, 0.00001}
Inner learning rate β	{0.001, 0.0005}
Weight decay ℓ_2	{0, 0.0001}
Inner gradient steps	{1, 3}
Max meta-test gradient steps	{20}
LSTM hidden dimension	{128, 256}
LSTM embedding sizes	{64, 128}
Batch size	16
Total meta-training environments	1000
Total meta-val/test environments	32
Maximum horizon (T)	15

A.1.2 *Environment Details*

The underlying MDP structure of SpriteWorld is a grid, where the states are each of the grid cells, and the actions enable the agent to move to any one of its 8-connected neighbors. The task visuals are inspired by Starcraft (e.g. Synnaeve et al., 2016), although we do not use the game engine. The sprites in our environment are extracted directly from the StarCraft files. We used in total 100 random units for meta-training. Evaluation on new objects was performed with 5 randomly selected sprites. For computational efficiency, we create a meta-training set of 1000 tasks and cache the optimal policy and state visitations under the true cost. Our evaluation is over 32 tasks. Our set of sprites was divided into two categories: buildings and characters. Each characters had multiple poses (taken from different frames of animation, such as walking/running/flying), whereas buildings only had a single pose. During meta-training the units were randomly placed, but to avoid the possibility that the agent would not need to actively avoid obstacles, the units were placed away from the boundary of the image in both the meta-validation and meta-test set.

The terrain in each environment was randomly generated using a set of tiles, each belonging to a specific category (e.g. grass, dirt, water). For each tile, we also specified a set of possible tiles for each of the 4-neighbors. Using these constraints on the neighbors, we generated random environment terrains using a graph traversal algorithm, where successor tiles were sampled randomly from this set of possible tiles. This process resulted in randomly generated, seamless environments. The expert demonstrations were generated using a cost (negative reward) of 8 for the obstacles, 2 for any grass tile, and 1 for any dirt tile. The names of the units used in our experiments are as follows (names are from the original game files):

The list of buildings used is: academy, assim, barrack, beacon, cerebrat, chemlab, chrysal, cocoon, comsat, control, depot, drydock, egg, extract, factory, fcolony, forge, gateway, genelab, geyser, hatchery, hive, infest, lair, larva, mutapit, nest, nexus, nukesilo, nydustpit, overlord, physics, probe, pylon, prism, pillbox, queen, rcluster, refinery, research, robotic, sbattery, scolony, spire, starbase, stargate, starport, temple, warm, weaponpl, wessel.

The list of characters used is: acritter, arbiter, archives, archon, avenger, battlecr, brood, bugguy, carrier, civilian, defiler, dragoon, drone, dropship, firebat, gencore, ghost, guardian, hydra, intercep, jcritter, lurker, marine, missile, mutacham, mutalid, sapper, scout, scv, shuttle, snakey, spider, stank, tank, templar, trilob, ucereb, uikerr, ultra, vulture, witness, zealot, zergling.

A.2 SUNCG EXPERIMENTAL DETAILS

A.2.1 *Algorithmic Details*

Table 7: Hyperparamters on the SUNCG environment. Curly brackets indicate that the the parameter was chosen from that set.

Hyperparameters	Value
Architecture	Conv($16-5\times5-1$)
	Conv($32-3\times3-1$)
	MLP(32)
	MLP(1)
Max number of training steps	15000000
Number of seed	3
Learning rate α	{0.1, 0.01, 0.001, 0.0001}
Inner learning rate β	{0.15, 0.1, 0.01, 0.0001}
Inner gradient steps	{3, 5}
Max meta-test gradient steps	{10}
Momentum	{0.9, 0.95, 0.99}

Our per task MaxEnt IRL baseline is learned by using the same base architecture. To provide a fair comparison, we do not use an inner learning rule in the inner loop of ManDRIL such as Adam Kingma and Ba, 2014 and use regular SGD. For our baseline however, we include a momentum term over which we tune. We tune over the number of training steps, learning rate and momentum parameters. We use SGD with momentum. For ManDRIL, we tune over the inner learning rate β and learning rate α and number of gradient steps. At meta-test time, we experimented with taking up to 10 gradient steps. For pretraining IRL, we first train for 150,000 steps, freeze the weights, and fine tune them for every separate task. For training from scratch, we use the Glorot uniform initialization in the the convolutional layers Glorot and Bengio, 2010.

A.2.2 *Environment Details*

Table 8: Summary of SUNCG environment setup.

Hyperparameters	Value
Discount (γ)	0.99
Maximum horizon (T)	40
Initial random steps	30
Number of demonstrations	5
Training environments	1004
Test environments	236
Test-house environments	173
(PICK/NAV) split:	716/697

The MDP in each environment is discretized into a grid where the state is defined by the grid coordinates plus the agent's orientation (N,S,E,W). The agent receives an observation which is a first-person panoramic view. The panoramic view consists of four 32×24 semantic image observations containing 61 channels.

The only departure for the task setup of Fu et al. (2019) that we make is to randomize the agent's start location by executing a random walk at the beginning of each episode. In Fu et al. (2019), the agent's start location was previously deterministic which allows a trivial solution of memorizing the provided demonstrations.

A.3 SPRITEWORLD META-TEST TRAINING PERFORMANCE

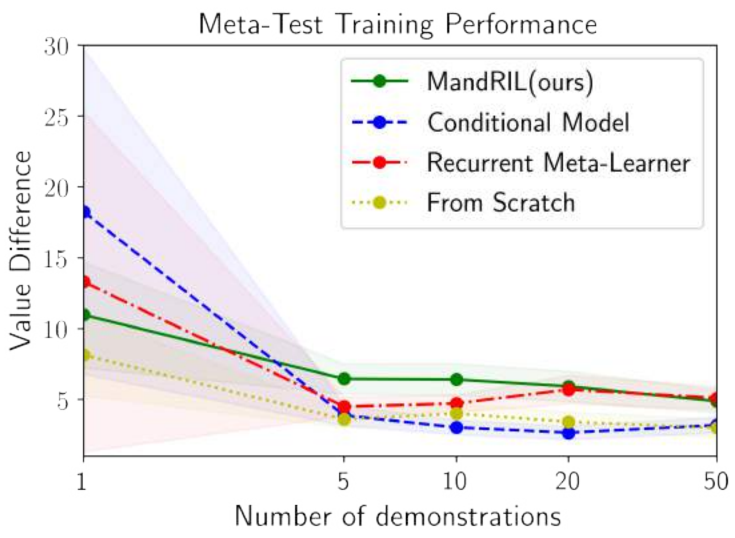


Figure 42: Meta-test "training" performance with varying numbers of demonstrations (lower is better). This is the performance on the environment for which demonstrations are provided for adaptation. As the number of demonstrations increase, all methods are able to perform well in terms of training performance as they can simply overfit to the training environment without acquiring the right visual cues that allow them to generalize. However, we find comes at the cost comes of considerable overfitting as we discuss in Section. 7.5.

A.4 SUNCG DAGGER PERFORMANCE

Table 9: DAgger success rate (%) on heldout tasks with 5 demonstrations. ManDRIL values are repeat for viewing convenience. Results are averaged over 3 random seeds.

Method	Test			Unseen Houses		
	Pick	NAV	Total	Pick	Nav	Total
DAgger	1.0	12.8	7.5	7.4	15.5	11.8
MandRIL(ours)	**52.3**	**90.7**	**77.3**	**56.3**	**91.0**	**82.6**

Here we show the performance of DAgger (Ross et al., 2011), in the setting where the number of samples that is equal to the number of demonstrations. Overall, while DAgger slightly improves performance over behavioral cloning, the performance still lags significantly behind ManDRIL and other IRL methods.

A.5 DETAILED META-OBJECTIVE DERIVATION

We define the quality of reward function r_θ parameterized by $\theta \in \mathbb{R}^k$ on task $\mathcal{T}$ with the MaxEnt IRL loss, $\mathcal{L}^{\mathcal{T}}_{\text{IRL}}(\theta)$, described in Section 2.3. The corresponding gradient is

$$\nabla_\theta \mathcal{L}_{\text{IRL}}(\theta) = \frac{\partial\, r_\theta}{\partial\, \theta}(\mathbb{E}_\tau[\mu_\tau] - \mu_{\mathcal{D}_\mathcal{T}}), \tag{21}$$

where $\partial\, r_\theta / \partial\, \theta$ is the $k \times |\mathcal{S}||\mathcal{A}|$-dimensional Jacobian matrix of the reward function r_θ with respect to the parameters θ. Here, $\mu_\tau \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ is the vector of *state-action visitations* under the trajectory τ (i.e. the vector whose elements are 1 if the corresponding state-action pair has been visited by the trajectory τ, and 0 otherwise), and $\mu_{\mathcal{D}_\mathcal{T}} = \frac{1}{|\mathcal{D}_\mathcal{T}|}\sum_{\tau \in \mathcal{D}_\mathcal{T}} \mu_\tau$ is the mean state visitations over all demonstrated trajectories in $\mathcal{D}_\mathcal{T}$. Let $\phi_\mathcal{T} \in \mathbb{R}^k$ be the updated parameters after a single gradient step. Then

$$\phi_\mathcal{T} = \theta - \alpha \nabla_\theta \mathcal{L}^{\text{tr}}_\mathcal{T}(\theta). \tag{22}$$

Let $\mathcal{L}^{\text{test}}_\mathcal{T}$ be the MaxEnt IRL loss, where the expectation over trajectories is computed with respect to a test set that is *disjoint* from the set of demonstrations used to compute $\mathcal{L}^{\text{test}}_\mathcal{T}(\theta)$ in Eq. 22. We seek to minimize

$$\sum_{\mathcal{T} \in \mathcal{T}^{\text{test}}} \mathcal{L}^{\text{test}}_\mathcal{T}(\phi_\mathcal{T}) \tag{23}$$

over the parameters θ. To do so, we first compute the gradient of Eq. 23, which we derive here. Applying the chain rule

$$\begin{aligned} \nabla_\theta \mathcal{L}^{\text{test}}_\mathcal{T} &= \frac{\partial\, \phi_\mathcal{T}}{\partial\, \theta} \frac{\partial\, r_{\phi_\mathcal{T}}}{\partial\, \phi_\mathcal{T}} \frac{\partial\, \mathcal{L}^{\text{test}}_\mathcal{T}}{\partial\, r_{\phi_\mathcal{T}}} \\ &= \frac{\partial}{\partial\, \theta}\left(\theta - \alpha \nabla_\theta \mathcal{L}^{\text{tr}}_\mathcal{T}(\theta)\right) \frac{\partial\, r_{\phi_\mathcal{T}}}{\partial\, \phi_\mathcal{T}} \frac{\partial\, \mathcal{L}^{\text{test}}_\mathcal{T}}{\partial\, r_{\phi_\mathcal{T}}} \\ &= \left(\mathbf{I} - \alpha \frac{\partial}{\partial\, \theta}\left(\frac{\partial\, r_\theta}{\partial\, \theta}(\mathbb{E}_\tau[\mu_\tau] - \mu_{\mathcal{D}_\mathcal{T}})\right)\right) \frac{\partial\, r_{\phi_\mathcal{T}}}{\partial\, \phi_\mathcal{T}} \frac{\partial\, \mathcal{L}^{\text{test}}_\mathcal{T}}{\partial\, r_{\phi_\mathcal{T}}} \end{aligned} \tag{24}$$

where in the last equation we substitute in the gradient of the MaxEnt IRL loss in Eq. 21 for $\nabla_\theta \mathcal{L}^{\text{tr}}_\mathcal{T}(\theta)$. In Eq. 24, we use the following notation:

- $\partial\, \phi_\mathcal{T} / \partial\, \theta$ denotes the $k \times k$-dimensional vector of partial derivatives $\partial\, \phi_{\mathcal{T},i} / \partial\, \theta_j$,
- $\partial\, r_{\phi_\mathcal{T}} / \partial\, \phi_\mathcal{T}$ denotes the $k \times |\mathcal{S}||\mathcal{A}|$-dimensional matrix of partial derivatives $\partial\, r_{\phi_{\mathcal{T},i}} / \partial\, \phi_{\mathcal{T},j}$,

- and, $\partial \mathcal{L}_{\mathcal{T}}^{\text{test}} / \partial r_{\phi_{\mathcal{T}}}$ denotes the k-dimensional gradient vector of $\mathcal{L}_{\mathcal{T}}^{\text{test}}$ with respect to $r_{\phi_{\mathcal{T}}}$.

We will now focus on the term inside of the parentheses in Eq. 24, which is a $k \times k$-dimensional matrix of partial derivatives.

$$\begin{aligned}
&\frac{\partial}{\partial \theta}\left(\frac{\partial r_\theta}{\partial \theta}(\mathbb{E}_\tau[\mu_\tau] - \mu_{\mathcal{D}_{\mathcal{T}}})\right) \\
&= \sum_{i=1}^{|\mathcal{S}||\mathcal{A}|}\left[\frac{\partial^2 r_\theta}{\partial \theta^2}(\mathbb{E}_\tau[\mu_\tau] - \mu_{\mathcal{D}_{\mathcal{T}}})_i + \frac{\partial}{\partial \theta}(\mathbb{E}_\tau[\mu_\tau])_i \left(\frac{\partial r_{\theta,i}}{\partial \theta}\right)^\top\right] \\
&= \sum_{i=1}^{|\mathcal{S}||\mathcal{A}|}\left[\frac{\partial^2 r_\theta}{\partial \theta^2}(\mathbb{E}_\tau[\mu_\tau] - \mu_{\mathcal{D}_{\mathcal{T}}})_i + \right. \\
&\left(\frac{\partial r_{\theta,i}}{\partial \theta}\right)\left(\frac{\partial}{\partial r_{\theta,i}}(\mathbb{E}_\tau[\mu_\tau])_i\right)\left(\frac{\partial r_{\theta,i}}{\partial \theta}\right)^\top]
\end{aligned}$$

where between the first and second lines, we apply the chain rule to expand the second term. In this expression, we make use of the following notation:

- $\partial^2 r_\theta / \partial \theta^2$ denotes the $k \times |\mathcal{S}||\mathcal{A}|$-dimensional matrix of second-order partial derivatives of the form $\partial^2 r_{\theta,i} / \partial \theta_j^2$,
- $(\mathbb{E}_\tau[\mu_\tau] - \mu_{\mathcal{D}_{\mathcal{T}}})_i$ denotes the ith element of the $|\mathcal{S}||\mathcal{A}|$-dimensional vector $(\mathbb{E}_\tau[\mu_\tau] - \mu_{\mathcal{D}_{\mathcal{T}}})_i$,
- $\partial r_{\theta,i} / \partial \theta$ denotes the k-dimensional matrix of partial derivatives of the form $\partial r_{\theta,i} / \partial \theta_j$ for $j = 1, 2, \dots, k$,
- and, $\frac{\partial}{\partial r_{\theta,i}}(\mathbb{E}_\tau[\mu_\tau])_i$ is the partial derivative of the ith element of the $|\mathcal{S}||\mathcal{A}|$-dimensional vector $\mathbb{E}_\tau[\mu_\tau]$ with respect to the ith element of the $|\mathcal{S}||\mathcal{A}|$-dimensional vector r_θ of reward (i.e. the reward function).

When substituted back into Eq. 24, the resulting gradient is equivalent to that in Eq. 8 in Section 2.3. In particular, defining the $|\mathcal{S}||\mathcal{A}|$-dimensional diagonal matrix D as

$$D := \operatorname{diag}\left(\left\{\frac{\partial}{\partial r_{\theta,i}}(\mathbb{E}_\tau[\mu_\tau])_i\right\}_{i=1}^{|\mathcal{S}||\mathcal{A}|}\right)$$

then the final term can be simplified to

$$\sum_{i=1}^{|\mathcal{S}||\mathcal{A}|} \left(\frac{\partial\, r_{\theta,i}}{\partial\, \theta}\right) \left(\frac{\partial}{\partial\, r_{\theta,i}} (\mathbb{E}_\tau[\boldsymbol{\mu}_\tau])_i\right) \left(\frac{\partial\, r_{\theta,i}}{\partial\, \theta}\right)^\top$$
$$= \left(\frac{\partial\, r_\theta}{\partial\, \theta}\right) D \left(\frac{\partial\, r_\theta}{\partial \theta}\right)^\top.$$

In order to compute this gradient, however, we must take the gradient of the expectation $\mathbb{E}_\tau[\boldsymbol{\mu}_\tau]$ with respect to the reward function r_θ. This can be done by expanding the expectation as follows

$$\begin{aligned}
\frac{\partial}{\partial\, r_\theta}\mathbb{E}_\tau[\boldsymbol{\mu}_\tau] &= \frac{\partial}{\partial\, r_\theta} \sum_\tau \left(\frac{\exp(\boldsymbol{\mu}_\tau^\top r_\theta)}{\sum_{\tau'} \exp(\boldsymbol{\mu}_{\tau'}^\top r_\theta)}\right) \boldsymbol{\mu}_\tau \\
&= \sum_\tau \left(\left(\frac{\exp(\boldsymbol{\mu}_\tau^\top r_\theta)}{\sum_{\tau'} \exp(\boldsymbol{\mu}_{\tau'}^\top r_\theta)}\right)(\boldsymbol{\mu}_\tau \boldsymbol{\mu}_\tau^\top) - \frac{\exp(\boldsymbol{\mu}_\tau^\top r_\theta)}{(\sum_{\tau'} \exp(\boldsymbol{\mu}_{\tau'}^\top r_\theta))^2} \sum_{\tau'} (\boldsymbol{\mu}_{\tau'} \boldsymbol{\mu}_\tau^\top) \exp(\boldsymbol{\mu}_{\tau'}^\top r_\theta)\right) \\
&= \sum_\tau P(\tau \mid r_\theta)(\boldsymbol{\mu}_\tau \boldsymbol{\mu}_\tau^\top) - \sum_\tau P(\tau|r_\theta) \sum_{\tau'} P(\tau' \mid r_\theta)(\boldsymbol{\mu}_{\tau'} \boldsymbol{\mu}_\tau^\top) \\
&= \mathbb{E}_\tau \left[(\boldsymbol{\mu}_\tau \boldsymbol{\mu}_\tau^\top) - \sum_{\tau'} P(\tau' \mid r_\theta)(\boldsymbol{\mu}_{\tau'} \boldsymbol{\mu}_\tau^\top)\right] \\
&= \mathbb{E}_\tau[\boldsymbol{\mu}_\tau \boldsymbol{\mu}_\tau^\top] - \mathbb{E}_{\tau',\tau}[\boldsymbol{\mu}_{\tau'} \boldsymbol{\mu}_\tau^\top] \\
&= \mathbb{E}_\tau[\boldsymbol{\mu}_\tau \boldsymbol{\mu}_\tau^\top] - \mathbb{E}_\tau[\boldsymbol{\mu}_\tau](\mathbb{E}_\tau[\boldsymbol{\mu}_\tau])^\top \\
&= \mathrm{Cov}[\boldsymbol{\mu}_\tau].
\end{aligned}$$

B

APPENDIX: PROBABILISTIC MODEL AGNOSTIC META-LEARNING

B.1 AMBIGUOUS CELEBA DETAILS

We construct our ambiguous few-shot variant of CelebA using the canonical splits to form the meta-train/val/test set. This gives us a split of 162770/19867/19962 images respectively. We additionally randomly partition the 40 available attributes and into a split of 25/5/10, which we use to construct the tasks below.

During training, each task is constructed by randomly sampling 2 attributes as Boolean variables and constructing tasks where one class shares the setting of these attributes and the other is the converse. For example, a valid constructed tasks is classifying `not Smiling, Pale Skin` versus `Smiling, not Pale Skin`. During testing, we sample 3 attributes from the test set to form the training task, and sample the 3 corresponding 2-uples to form the test task. After removing combinations that have insufficient examples to form a single tasks, this scheme produces 583/19/53 tasks for meta-train/val/test respectively. Each sampled image is pre-processed by first obtaining an approximately 168×168 center crop of each image following by downsampling to 84×84. This crop is captures regions of the image necessary to classify the non-facial attributes (e.g. Wearing Necklace).

Meta-training attributes:

`Oval Face, Attractive, Mustache, Male, Pointy Nose, Bushy Eyebrows, Blond Hair, Rosy Cheeks, Receding Hairline, Eyeglasses, Goatee, Brown Hair, Narrow Eyes, Chubby, Big Lips, Wavy Hair, Bags Under Eyes, Arched Eyebrows, Wearing Earrings, High Cheekbones, Black Hair, Bangs, Wearing Lipstick, Sideburns, Bald`

Meta-validation attributes:

`Wearing Necklace, Smiling, Pale Skin, Wearing Necktie, Big Nose`

Meta-testing attributes:

`Straight Hair, 5 o'Clock Shadow, Wearing Hat, Gray Hair, Heavy Makeup, Young, Blurry, Double Chin, Mouth Slightly Open, No Beard.`

B.2 EXPERIMENTAL DETAILS

In the illustrative experiments, we use a fully connected network with 3 ReLU layers of size 100. Following Finn et al. (2017d), we additionally use a bias transformation variable, concatenated to the input, with size 20. Both methods use 5 inner gradient steps on $\mathcal{D}^{\text{tr}}$ with step size $\alpha = 0.001$ for regression and $\alpha = 0.01$ for classification. The inference network and prior for PLATIPUS both use one gradient step. For PLATIPUS, we weight the KL term in the objective by 1.5 for 1D regression and 0.01 for 2D classification.

For CelebA, we adapt the base convolutional architecture described in Finn et al. (2017a) which we refer the readers to for more detail. Our approximate posterior and prior have dimensionality matching the underlying model. We tune our approach over the inner learning rate α, a weight on the D_{KL}, the scale of the initialization of $\boldsymbol{\sigma}^2_\theta, \mathbf{v}_q \in \{0.5, 0.1, 0.15\}$, $\boldsymbol{\gamma}_p, \boldsymbol{\gamma}_q \in \{0.05, 0.1\}$, and a weight on the KL objective $\in \{0.05, 0.1, 0.15\}$ which we anneal towards during training. All models are trained for a maximum of 60,000 iterations.

At meta-test time, we evaluate our approach by taking 15 samples from the prior before determining the assignments. The assignments are made based on the likelihood of the testing examples. We average our results over 100 test tasks. In order to compute the marginal log-likelihood, we average over 100 samples from the prior.

B.3 MINIIMAGENET COMPARISON

We provide an additional comparison on the MiniImagenet dataset. Since this benchmark does not contain a large amount of ambiguity, we do not aim to show state-of-the-art performance. Instead, our goal with this experiment is to compare our approach on to MAML and prior methods that build upon MAML on this standard benchmark. Since our goal is to compare algorithms, rather than achieving maximal performance, we decouple the effect of the meta-learning algorithm and the architecture used by using the standard 4-block convolutional architecture used by Vinyals et al. (2016); Ravi and Larochelle (2017); Finn et al. (2017a) and others. We note that better performance can likely be achieved by tuning the architecture. The results, in Table 10 indicate that our method slightly outperforms MAML and achieves comparable performance to a number of other prior methods.

MiniImagenet	5-way, 1-shot Accuracy
MAML Finn and Levine, 2017	48.70 ± 1.84%
LLAMA Grant et al., 2018b	49.40 ± 1.83%
Reptile Nichol and Schulman, 2018	**49.97 ± 0.32**%
PLATIPUS (ours)	**50.13 ± 1.86**%
Meta-SGD Li et al., 2017b	**50.71 ± 1.87**%
matching nets Vinyals et al., 2016	43.56 ± 0.84%
meta-learner LSTM Ravi and Larochelle, 2017	43.44 ± 0.77%
SNAIL Mishra et al., 2018*	45.10 ± 0.00%
prototypical networks Snell et al., 2017	46.61 ± 0.78%
mAP-DLM Snell et al., 2017	49.82 ± 0.78%
GNN Garcia and Bruna, 2017	**50.33 ± 0.36**%
Relation Net Sung et al., 2017	**50.44 ± 0.82**%

Table 10: Comparison between our approach and prior MAML-based methods (top), and other prior few-shot learning techniques on the 5-way, 1-shot MiniImagenet benchmark. Our approach gives a small boost over MAML, and is comparable to other approaches. We bold the approaches that are above the highest confidence interval lower-bound. *Accuracy using comparable network architecture.

C

APPENDIX: LEARNING SKILLS FROM RESETS

We first describe the details of environments used in the paper, and next list the hyper-parameters used to train all the agents. All environments use the MuJoCo 2.0 simulator.

C.1 ENVIRONMENT DETAILS

C.1.1 *Ant Task*

The `Ant` environment is equivalent to the standard gym `Ant-v3` environment except that the gear ratio is reduced from $(-120, 120)$ to $(-30, 30)$. The use of this lower gear ratio is consistent with prior work Eysenbach et al., 2018. The observation space is the environment $qpos$ and $qvel$. The rewards are a weighted combination of three terms: (1) a negative reward of the distance from the center, (2) a positive reward for reaching a threshold from the center and (3). The equation used is for (1) and (2) is given below:

$$r(d) = e^{-d^2/2} + \text{num goals completed}$$

where d is the $\ell - 2$ norm of the distance from the origin. A negative reward (−300) is also given to the agent for flipping over. If the reset policy terminates the episode in this manner then the forward policy is not executed. If however, the forward policy terminates the episode, the forward policy receives the flip cost and the reset policy includes the negation of this cost in its reward as described in Sec. 4.3.2. Each agent is given a time horizon of $T_{\text{reset}} = T = 200$. In this ant task, the game was run for 7000 episodes.

C.1.2 *Ant-Waypoints Task*

The aim of this task is to navigate between a set of pre-defined waypoints. The waypoints defined are $[(0,5),(5,5),(5,0)]$. The reward structure is the same as the constructed ant environment described in section E.2.1, conditioned on the current waypoint.

The lower level controller was constructed from the reset and forward skills learned from the adversarial game. The (x,y) position of the state were re-normalized so that the ant appeared to be at the origin. The higher level controller was trained using an implementation of Double DQN Van Hasselt et al., 2016 using the 2-D coordinates of the Ant as the state.

C.1.3 *Ant-MediumMaze Task*

We use the base environment from D4RL using a single goal[1] Fu et al., 2020. The reward is the $\ell - 2$ distance to the target position. The target position is the top right corner and the bottom position is the bottom left corner. Each episode is of a length 30 where each step represents a single skill.

C.1.4 *ManipulateFreeObject Task*

The object is a 6 DoF three pronged object that is 15cm in diameter placed in a workspace that is 30 cm $\times$ 30 cm box. We follow the goal configuration of Zhu et al. (2019), using $(x,y,\theta) = (0,0,-\frac{\pi}{2})$, and use the same 15 initial configurations used for evaluation. Experiments were performed using a state representation that comprised of the object (x,y) position, and sin and cos encoding of its orientation, the claw's pose, and the last action taken.

1 https://github.com/rail-berkeley/d4rl

C.2 ADDITIONAL EXPERIMENTS

C.2.1 *Understanding the effects of resets and state representation choice on Ant-Waypoints*

Here, we provide an analysis of the effects of resets and state representation choice on downstream task performance on the `Ant-Waypoints` task. We vary the condition under which the policy is learned (i.e., reset or reset-free) and the state representation (full state versus (x, y)-prior). In general, we find that only with the availability of the (x, y)-prior and resets are prior methods able to make significant progress (Fig. 45). The different relative performance of the same skills on different downstream tasks (Waypoints vs Maze) also suggests that learning a single action representation may not always be ideal. This suggests future work that combines downstream adaptation, or potentially mixing in primitive actions to allow for more granular control.

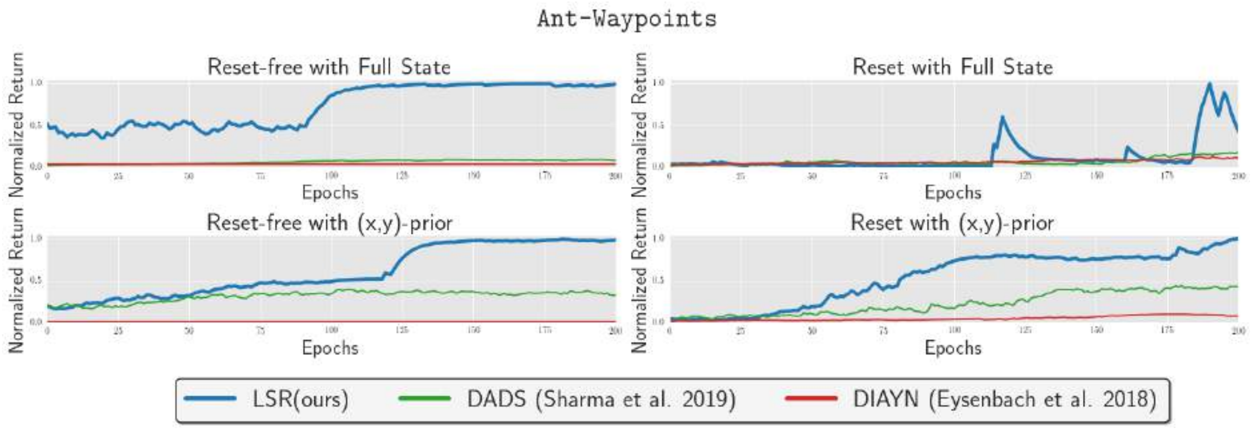


Figure 43: A evaluation of the skills learned when varying the state representation ((x, y)-prior or full state) and initial state distribution (reset-free or with resets) on the `Ant-Waypoints` task. We normalize the return of plot in terms of the best performing algorithm. Similar to the `Ant-MediumMaze` task, we find that prior methods can only make progress on the task when resets or additional information in the form of the (x, y)-prior is available.

C.2.2 *Effect of varying the number of skills on downstream performance*

C.3 ALGORITHM DETAILS

The adversarial game comprises of a reset policy and a forward policy as described in the paper. The forward policy receives rewards as given by the environment at every

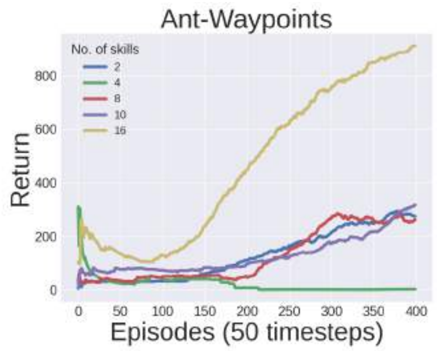


Figure 44: A plot showing the effect of varying the number of skills on the `Ant-Waypoints` tasks. We find that increasing the number of skills improves performance (16, yellow curve), the value used in our experiments (10, purple curve) is far from the best, and many other values attain similar performance. The number of skills used in our experiments were chosen to be roughly around the order of previous work Eysenbach et al., 2018.

step. The reset policy receives rewards from DIAYN at every step and a game reward at the last step. This game reward is the sum of the rewards received by the forward policy when played from the final state of the reset policy trajectory. Each agent is given a fixed number of steps at each turn. Environment termination was handled for individually for both agents as follows. If the reset policy terminates, the forward policy does not take any steps in the environment and is not given any reward. If forward policy terminates, the reset policy is given rewards based on the forward policy as normal.

In the following sections, curly brackets indicate that a parameter was searched over.

C.3.1 *DClaw Environment Parameters*

The experiments conducted on the `DClaw-ManipulateFreeObject` domain was conducted using the following parameters. The grid search parameters for DIAYN were identical to those used in our method except where $\lambda = 0$.

Hyperparameters	**Value**
Actor-critic architecture	FC(256, 256)
RND architecture	FC(256, 256, 512)
Classifier architecture	FC(256, 256)
Optimizer	Adam
Learning rate	{3e-3, 3e-4}
Classifier steps per iteration	5
γ	0.99
λ	{0.1, 0.5}
r_{skill} scale	0.1
target update	0.005
Batch size	256
Classifier batch size	128

We follow Zhu et al., 2020a and provide 200 goal images for all experiments which are used to learn a VICE Fu et al., 2018b based reward function. We similarly omit the $-\log \pi(\mathbf{s}|\mathbf{a})$. We also follow Burda et al. (2018) and normalize the prediction errors. In our implementation of LSR, we found used a dimensionality of $\mathbf{z}$ of 2, implemented as a separate network as opposed to one network. The reset controller Eysenbach et al., 2017 and R3L baseline Zhu et al., 2020a follow the hyperparameters above. The reset controller's results were averaged over the 3 positions suggested by Zhu et al. (2020a).

C.3.2 *Ant Environment Parameters*

The experiments conducted on the `Ant` domain were conducted using the following parameters. The grid search parameters for DIAYN were identical to those used in our method except where $\lambda = 0$.

Hyperparameters	**Value**
Actor-critic architecture	FC(256, 256)
Classifier architecture	FC(256, 256)
Optimizer	Adam
Learning rate	{1e-3, 2e-4}
Training steps per agent	$\{200, 200\}$
γ	0.99
Episode Length	1000
λ	0.5
Number of skills	10
r_{skill} scale	$[0.01, 10]$, chose 2
target update	0.005
Batch size	256
Classifier batch size	256

C.3.3 *Hierarchical Controller Parameters*

The hierarchical controller is a Double-DQN Van Hasselt et al., 2016 network that takes as input the $\mathbf{s}_t$ and outputs a Q-function over the skills $Q(\mathbf{s}_t, \mathbf{z})$.

Hyperparameters	**Value**
Architecture	FC(128)
Exploration fraction	0.3
Final epsilon j	0.0
Learning rate α	0.0001
Reward Scale	{0.1, 0.3}
Batch size	256
Maximum horizon (T)	30
Total epochs	500
Replay Buffer size	{1000, 10,000}
Critic update frequency	{5, 10}
Critic updates per epoch	{100, 200}

An episode step corresponds to a 150 length rollout by the lower level controller in the environment.

C.3.4 *DADS*

To allow for fair comparison, we run DADS without an (x, y) prior. We additionally discretize the continuous skill space into a space of 10 discrete skills to allow for us to use the same meta-controller architecture. We used the open source implementation of DADS Sharma et al., 2019 [2] with the following parameters:

Skill parameters	
Number of skills	10
Skill type	Discrete
Dynamics parameters	
train steps	8
learning rate	3e-4
batch size	256
Path length	200
Agent parameters	
Architecture	FC(512)
Learning rate	3e-4
Entropy	0.1
Train steps	64
Batch size	256
γ	0.99

2 https://github.com/google-research/dads

D

APPENDIX: MULTI-TASK RESET-FREE LEARNING FOR DEXTEROUS MANIPULATION

We first describe the details of environments used in the paper, and next list the hyper-parameters used to train all the agents. All environments use the MuJoCo 2.0 simulator.

D.1 ENVIRONMENT DETAILS

D.1.1 *Ant Task*

The `Ant` environment is equivalent to the standard gym `Ant-v3` environment except that the gear ratio is reduced from $(-120, 120)$ to $(-30, 30)$. The use of this lower gear ratio is consistent with prior work Eysenbach et al., 2018. The observation space is the environment $qpos$ and $qvel$. The rewards are a weighted combination of three terms: (1) a negative reward of the distance from the center, (2) a positive reward for reaching a threshold from the center and (3). The equation used is for (1) and (2) is given below:

$$r(d) = e^{-d^2/2} + \text{num goals completed}$$

where d is the $\ell - 2$ norm of the distance from the origin. A negative reward (−300) is also given to the agent for flipping over. If the reset policy terminates the episode in this manner then the forward policy is not executed. If however, the forward policy terminates the episode, the forward policy receives the flip cost and the reset policy includes the negation of this cost in its reward as described in Sec. 4.3.2. Each agent is given a time horizon of $T_{\text{reset}} = T = 200$. In this ant task, the game was run for 7000 episodes.

D.1.2 *Ant-Waypoints Task*

The aim of this task is to navigate between a set of pre-defined waypoints. The waypoints defined are $[(0,5),(5,5),(5,0)]$. The reward structure is the same as the constructed ant environment described in section E.2.1, conditioned on the current waypoint.

The lower level controller was constructed from the reset and forward skills learned from the adversarial game. The (x,y) position of the state were re-normalized so that the ant appeared to be at the origin. The higher level controller was trained using an implementation of Double DQN Van Hasselt et al., 2016 using the 2-D coordinates of the Ant as the state.

D.1.3 *Ant-MediumMaze Task*

We use the base environment from D4RL using a single goal[1] Fu et al., 2020. The reward is the $\ell - 2$ distance to the target position. The target position is the top right corner and the bottom position is the bottom left corner. Each episode is of a length 30 where each step represents a single skill.

D.1.4 *ManipulateFreeObject Task*

The object is a 6 DoF three pronged object that is 15cm in diameter placed in a workspace that is 30 cm $\times$ 30 cm box. We follow the goal configuration of Zhu et al. (2019), using $(x,y,\theta) = (0,0,-\frac{\pi}{2})$, and use the same 15 initial configurations used for evaluation. Experiments were performed using a state representation that comprised of the object (x,y) position, and sin and cos encoding of its orientation, the claw's pose, and the last action taken.

1 https://github.com/rail-berkeley/d4rl

D.2 ADDITIONAL EXPERIMENTS

D.2.1 *Understanding the effects of resets and state representation choice on Ant-Waypoints*

Here, we provide an analysis of the effects of resets and state representation choice on downstream task performance on the `Ant-Waypoints` task. We vary the condition under which the policy is learned (i.e., reset or reset-free) and the state representation (full state versus (x, y)-prior). In general, we find that only with the availability of the (x, y)-prior and resets are prior methods able to make significant progress (Fig. 45). The different relative performance of the same skills on different downstream tasks (Waypoints vs Maze) also suggests that learning a single action representation may not always be ideal. This suggests future work that combines downstream adaptation, or potentially mixing in primitive actions to allow for more granular control.

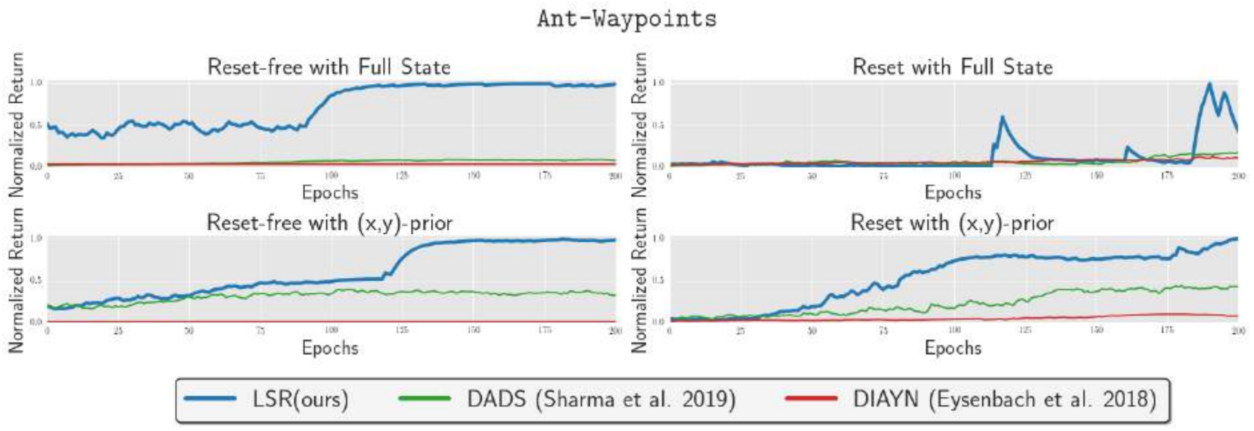


Figure 45: A evaluation of the skills learned when varying the state representation ((x, y)-prior or full state) and initial state distribution (reset-free or with resets) on the `Ant-Waypoints` task. We normalize the return of plot in terms of the best performing algorithm. Similar to the `Ant-MediumMaze` task, we find that prior methods can only make progress on the task when resets or additional information in the form of the (x, y)-prior is available.

D.2.2 *Effect of varying the number of skills on downstream performance*

D.3 ALGORITHM DETAILS

The adversarial game comprises of a reset policy and a forward policy as described in the paper. The forward policy receives rewards as given by the environment at every

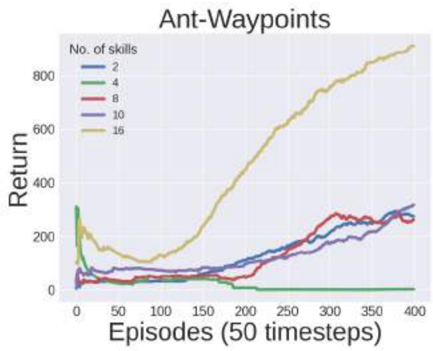


Figure 46: A plot showing the effect of varying the number of skills on the `Ant-Waypoints` tasks. We find that increasing the number of skills improves performance (16, yellow curve), the value used in our experiments (10, purple curve) is far from the best, and many other values attain similar performance. The number of skills used in our experiments were chosen to be roughly around the order of previous work Eysenbach et al., 2018.

step. The reset policy receives rewards from DIAYN at every step and a game reward at the last step. This game reward is the sum of the rewards received by the forward policy when played from the final state of the reset policy trajectory. Each agent is given a fixed number of steps at each turn. Environment termination was handled for individually for both agents as follows. If the reset policy terminates, the forward policy does not take any steps in the environment and is not given any reward. If forward policy terminates, the reset policy is given rewards based on the forward policy as normal.

In the following sections, curly brackets indicate that a parameter was searched over.

D.3.1 *DClaw Environment Parameters*

The experiments conducted on the `DClaw-ManipulateFreeObject` domain was conducted using the following parameters. The grid search parameters for DIAYN were identical to those used in our method except where $\lambda = 0$.

Hyperparameters	**Value**
Actor-critic architecture	FC(256, 256)
RND architecture	FC(256, 256, 512)
Classifier architecture	FC(256, 256)
Optimizer	Adam
Learning rate	{3e-3, 3e-4}
Classifier steps per iteration	5
γ	0.99
λ	{0.1, 0.5}
r_{skill} scale	0.1
target update	0.005
Batch size	256
Classifier batch size	128

We follow Zhu et al., 2020a and provide 200 goal images for all experiments which are used to learn a VICE Fu et al., 2018b based reward function. We similarly omit the $-\log \pi(\mathbf{s}|\mathbf{a})$. We also follow Burda et al. (2018) and normalize the prediction errors. In our implementation of LSR, we found used a dimensionality of $\mathbf{z}$ of 2, implemented as a separate network as opposed to one network. The reset controller Eysenbach et al., 2017 and R3L baseline Zhu et al., 2020a follow the hyperparameters above. The reset controller's results were averaged over the 3 positions suggested by Zhu et al. (2020a).

D.3.2 *Ant Environment Parameters*

The experiments conducted on the `Ant` domain were conducted using the following parameters. The grid search parameters for DIAYN were identical to those used in our method except where $\lambda = 0$.

Hyperparameters	**Value**
Actor-critic architecture	FC(256, 256)
Classifier architecture	FC(256, 256)
Optimizer	Adam
Learning rate	{1e-3, 2e-4}
Training steps per agent	$\{200, 200\}$
γ	0.99
Episode Length	1000
λ	0.5
Number of skills	10
r_{skill} scale	$[0.01, 10]$, chose 2
target update	0.005
Batch size	256
Classifier batch size	256

D.3.3 *Hierarchical Controller Parameters*

The hierarchical controller is a Double-DQN Van Hasselt et al., 2016 network that takes as input the $\mathbf{s}_t$ and outputs a Q-function over the skills $Q(\mathbf{s}_t, \mathbf{z})$.

Hyperparameters	**Value**
Architecture	FC(128)
Exploration fraction	0.3
Final epsilon j	0.0
Learning rate α	0.0001
Reward Scale	{0.1, 0.3}
Batch size	256
Maximum horizon (T)	30
Total epochs	500
Replay Buffer size	{1000, 10,000}
Critic update frequency	{5, 10}
Critic updates per epoch	{100, 200}

An episode step corresponds to a 150 length rollout by the lower level controller in the environment.

D.3.4 *DADS*

To allow for fair comparison, we run DADS without an (x, y) prior. We additionally discretize the continuous skill space into a space of 10 discrete skills to allow for us to use the same meta-controller architecture. We used the open source implementation of DADS Sharma et al., 2019 [2] with the following parameters:

Skill parameters	
Number of skills	10
Skill type	Discrete
Dynamics parameters	
train steps	8
learning rate	3e-4
batch size	256
Path length	200
Agent parameters	
Architecture	FC(512)
Learning rate	3e-4
Entropy	0.1
Train steps	64
Batch size	256
γ	0.99

2 https://github.com/google-research/dads

E

APPENDIX: LEARNING MULTI-TASK RESET-FREE BEHAVIOR FROM IMAGES

E.1 REAL WORLD ENVIRONMENT DESCRIPTIONS

In the following sections, we describe the details of our real world tasks. We provide details related to experimental setup and describe our success criteria. Finally, we describe the supervision we provide the agent.

E.1.1 *Object, Arena Dimensions, and Safety Considerations*

The objects used in our manipulation task are a 3-D printed pipe and hook that were custom designed. Their dimensions are shown in the technical drawing in Fig. 47. All the objects are manipulated in an arena of overall size 33" $\times$ 33" consisting of a base of 20" $\times$ 20" and 8" $\times$ 8" panels. Importantly, the fixtures we use, which can be seen in the example milestone images below, are made of a flexible foam. This is in order to ensure that the robot does not place excessive forces on either the object, hand, or fixture. We leave addressing these safety considerations to future work.

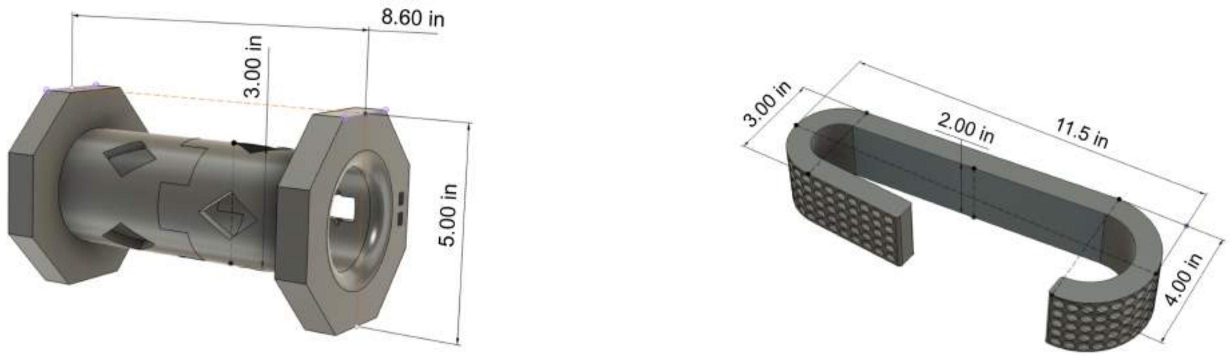


Figure 47: Technical drawing of objects (Left: hose connector, Right: hook) used for real-world experiments, with dimensions in inches.

E.1.2 *Hose Connector Insertion Task & Evaluation Criteria*

The goal of this task is to have the hand reach, grasp, reorient, and insert a hose connector into an insertion point. The hose connector is attached to a fixed point at the top of the 20in $\times$ 20in arena by a rope that is 31cm long.

For this environment, we collected 300 milestone images ($84 \times 84 \times 6$) for each subtask. When training the VICE classifiers, we apply data augmentation using a random crop on the provided milestone images in addition to a randomly sampled $\mathcal{N}(0, 0.02)$ noise vector on the state.

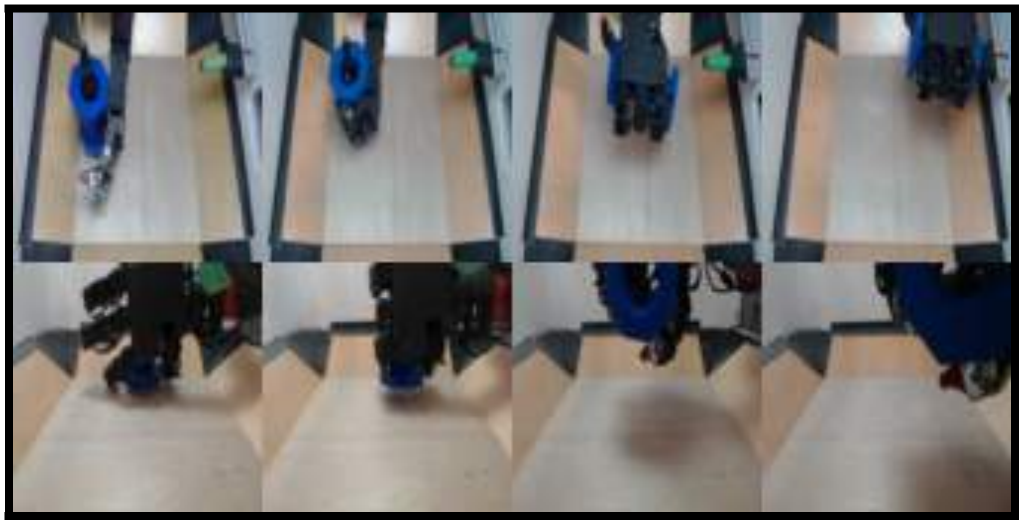


Figure 48: Sample milestone images for Reach, Grasp, Flipup, Insert (each column respectively))

In the following table, we summarize the success criteria that we use in our experiments.

Task	**Success Criteria**
Reach	$\mathbb{1}\left\{\lvert x_{palm} - x_{hose}\rvert < 0.05 \text{ AND } \lvert y_{palm} - y_{hose}\rvert < 0.01\right\}$
Grasp	$\mathbb{1}\left\{\text{ is\_held\_during\_flipup }\right\}$
FlipUp	$\mathbb{1}\left\{\lvert\theta_{hose} - \theta_{goal}\rvert \leqslant 5^\circ\right\}$
Insert	$\mathbb{1}\left\{\lvert x_{hose} - x_{peg}\rvert < 0.05 \text{ AND } \lvert y_{hose} - y_{peg}\rvert < 0.03\right\}$

In this environment, the evaluation criteria for successful finger grasps is intuitively defined as whether or not the grasp is firm enough for performing subsequent tasks, i.e. the hose connector does not fall from the hand. θ_{hose} is the Euler angle measurement of the hose connector, where the optimal insertion angle is $\theta_{goal} = 90^\circ$. x_{hose}, x_{peg}, y_{hose}, y_{peg} are the center of mass xy-coordinate of the hose connector and the insertion peg, measured in meters.

E.1.3 *Rope Hooking Task & Evaluation Criteria*

The goal of this task is to have the hand grasp, reorient, hook, and unhook a carabiner hook onto a latch. The hook is attached to a fixed point at the top of the arena by a rope that is 31cm long.

For this environment, we also collect 300 milestone images ($84 \times 84 \times 6$) for each subtask. Similar to the insertion task, when training the VICE classifiers, a random crop is used as data augmentation for the milestone images and a random $\mathcal{N}(0, 0.02)$ noise is added to the milestone states.

Task	**Success Criteria**
Grasp	$\mathbb{1}\left\{\text{ is\_held\_during\_flipup }\right\}$
FlipUp	$\mathbb{1}\left\{\lvert\theta_{hook} - \theta_{goal}\rvert \leqslant 5^\circ\right\}$
Hook	$\mathbb{1}\left\{(x_{hook} - x_{latch}) > 0.01\right\}$
Remove	$\mathbb{1}\left\{(x_{hook} - x_{latch}) < 0.05\right\}$

Similar to the hose insertion task, in this environment, the evaluation criteria for successful finger grasps is intuitively defined as whether the grasp is firm enough for performing subsequent tasks, i.e. the hook object does not fall from the hand. θ_{hook} is the Euler angle measurement of the hook object along the side of its flat handle, where the ready-to-hook angle is $\theta_{goal} = 90^\circ$. x_{hook} and x_{latch} is the center of mass x-coordinate of the hook object and the latching bar, measured in meters.

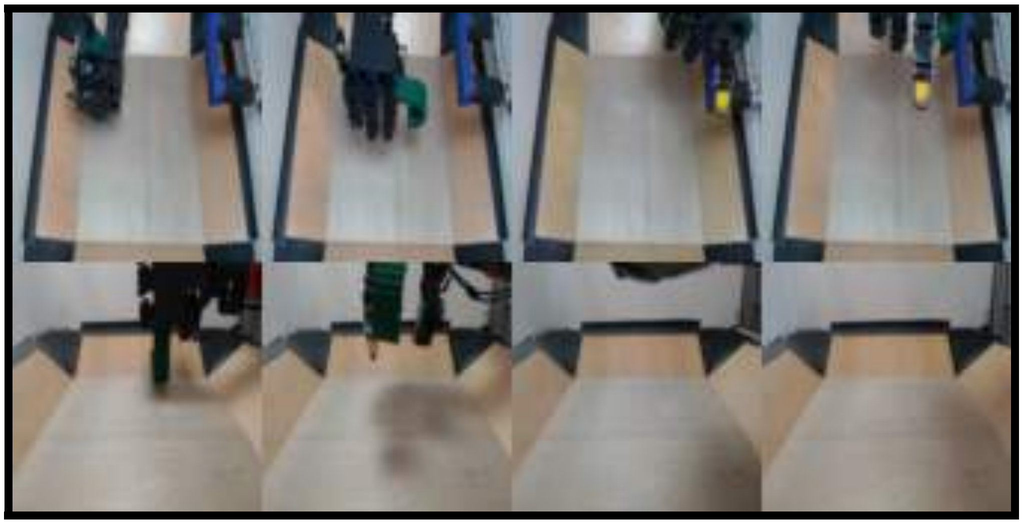


Figure 49: Sample milestone images for Grasp, Flipup, Hook, Unhook task, (each column respectively)

E.2 SIMULATED ENVIRONMENT DESCRIPTIONS

We first describe the details of simulated environments used in the paper, and next list the hyperparameters used to train all the agents. The environment we use is based on the MuJoCo 2.0 simulator.

E.2.1 *Valve3 Task*

The `DHandValve3` environment contains a green, three-pronged valve placed on top of a square arena with dimensions 0.55m x 0.55m. The valve contains a circular center, and each of its prongs has an equal length of 0.1m. There are three phases in this task: reach, reposition, and pickup.The reach phase's success criteria is when the hand is within 0.1m from the valve. In the reposition phase, the success criteria for the hand is to reach for the valve, grasp the valve with its fingers, and drag the valve to within 0.1m of the arena center. Finally, success of the pickup phase is measured if the object is picked up and brought within 0.1 m of the target location which is 0.2m above the table. In order to prevent the object from falling off the table, the object is constrained to a 0.15m radius by a string from the center of the table.

The observation space of the environment is two camera views of the robot, resized to $84 \times 84 \times 3$. In addition, the proprioceptive state of the arm is provided, which is comprised of a 16-dim hand joint position, 7-dim Sawyer arm state, and 6-dim vector representing the end-effector position and euler angle. The time horizon we use in each phase of the environment is $T = 100$. We provide 300 goal images per phase, which is comparable to the number used in prior work Fu et al., 2018b.

Below, we provide the oracle task graph for the valve environment, which we use to evaluate the relative performance of our learned task graph model.

Algorithm 6 Valve3 Task Graph (Oracle Baseline)

```
Require: Object position $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$, previous task $\phi$
 1: Let $\begin{bmatrix} x_{center} \\ y_{center} \end{bmatrix}$ be the center coordinates of the arena (relative to the Sawyer base).
 2: Let $\begin{bmatrix} x_{hand} \\ y_{hand} \end{bmatrix}$ be the location coordinates of the hand (relative to the arena).
 3: $\text{is\_centered} = \| \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_{center} \\ y_{center} \end{bmatrix} \| < 0.1$
 4: $\text{is\_hand\_over\_object} = \| \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x\_hand \\ y\_hand \end{bmatrix} \| < 0.15$
 5: if NOT is_centered and is_hand_over_object then
 6:     Reposition
 7: else if NOT is_centered and NOT is_hand_over_object then
 8:     Reach
 9: else if is_centered then
10:     Pickup
```

E.3 ALGORITHM DETAILS

In this section, we describe details related to our RL learning algorithms and also provide hyperparameters for each method.

E.3.1 *AVAIL*

Algorithm 7 AVAIL (Autonomy ViA mILestones)

```
1: Given: K tasks with examples states $\mathbf{D} := \{D_z, y_z\}_{z=0}^{K-1}$, start state $s_0$.
2: Train task graph $p_{taskdyn}(z|s)$ using $\mathbf{D}$
3: Initialize $\pi_z$, $p_z^o(o|s)$, $Q_z$, $\mathcal{B}_z$ for $z \in \{0, 1, \dots, K-1\}$
4: for iteration $n = 1, 2, \ldots$ do
5:     Select current task $z$ by querying learned task graph at the current state: $z = \text{argmax}_z p(z|s)$
6:     for iteration $j = 1, 2, \ldots, T$ do
7:         Execute $\pi_z$ in environment, storing data in the buffer $\mathcal{B}_z$
8:         Update the current task's policy and value functions $\pi_z$, $Q_z$ using samples from $\mathcal{B}_z$, assigning reward based on $p_z^o(o|s)$ using SAC Haarnoja et al., 2018c.
9:         Update the classifier parameters, using $D_z$ and samples from $\mathcal{B}_z$ , using the VICE Fu et al., 2018b.
```

E.3.2 *Reinforcement Learning from Images*

For completeness, here we describe our procedure of performing image based RL, which, as noted in prior work, presents significant optimization challenges Laskin et al., 2020; Kostrikov et al., 2020. In order to make learning more practical, we make use of a combination of data augmentation techniques during training, which has been previously shown to improve image based reinforcement learning Kostrikov et al., 2020, and dropout regularization (Hiraoka et al., 2021). For all approaches we evaluate on in this work we make use of random shifts perturbations, which pad the image observation with boundary pixels before taking a random crop. We denote $s_{aug} \sim f(s)$ as an randomly augmented image from a distribution f. We compute Q-Learning by computing the Q value for a state (s_i) over M independent augmentation. For each q function, $Q_\theta(s, a)$, we follow Hiraoka et al. (2021) and apply dropout followed by layer normalization in the fully connected layers of the critic.

$$\mathbb{E}_{\substack{s_i \sim B \\ a \sim \pi(\cdot|s)}}[Q_\theta(s, a)] \approx \frac{1}{M} \sum_{m=1}^{M} Q_\theta(f(s_i), a_m)$$
$$\text{where } a_m \sim \pi(\cdot|f(s_i)),$$

and computing a target value over L augmentations

$$y_i = r_i + \gamma \frac{1}{L} \sum_{l=1}^{L} Q_\theta(f(s_i', \nu_{i,l}'), a_{i,l}') \tag{25}$$
$$\text{where } a_{i,l}' \sim \pi(\cdot|f(s_i', \nu_{i,l}')). \tag{26}$$

This leads to a final learning rule

$$\theta \leftarrow \theta - \lambda_\theta \nabla_\theta \frac{1}{N} \sum_{i=1}^{N} (Q_\theta(f(s_i, \nu_i), a_i) - y_i)^2. \tag{27}$$

E.3.3 *Hyperparameters*

Here we describe the individual prior methods we compare to in detail for the purpose of reproducibility. For shared parameters, we summarize them below and provide baseline specific parameters separately.

For the simulated experiments, we train a classifier with an identical architecture to our success classifiers. During training we sample a new task a horizon of $T = 100$, as we found that in simulation the simpler naïve task graph performs comparably to the learned task graph, in real world training we employ the naïve task graph as it is arguably simpler. We provide additional experiments in the following section using the learned task which demonstrates that the learned task graph actually outperforms the naïve task graph on the more challenging real world domains.

Shared RL Hyperparameters	**Value**
Base Encoder	Conv(3, 3, 32, 2)
	3 $\times$ Conv(3, 3, 32, 1)
Actor Architecture	FC(256, 256)
	FC(256, 22)
Critic Architecture	FC(256, 256)
	FC(256, 1)
Optimizer	Adam
Learning rate	{3e-4}
Discount γ	0.99
Target Update Frequency	1
Actor Update Frequency	1
Batch size	256
Classifier batch size	256

Shared Classifier Hyperparameters	**Value**
Optimizer	Adam
Learning rate	{3e-4}
Classifier steps per iteration	1
Mixup Augmentation α	10
Label Smoothing α	0.1
Classifier Architecture	Conv(3, 3, 32, 2)
	$3 \times$ Conv(3, 3, 32, 1)
	$3 \times$ FC(512) $\rightarrow$ ReLU() $\rightarrow$ Dropout(0.5)
	FC(1)

E.4 ADDITIONAL REAL WORLD COMPARISONS

We additionally provide real world comparisons that mirror the more extensive comparisons done in simulation. When compared to prior methods, we find that our simulated results are corroborated by our real world experiments. We find that our approach outperforms the forward backward algorithm on our hose insertion and hook tasks. In evaluating the efficacy of our learned task, we find, however, that on the more challenging real world tasks, our method substantially performs a naíve task graph. We provide additional details here.

E.4.1 *Real World Comparisons to the Forward-Backward Controller (Eysenbach et al., 2017)*

In real world setting, we experiment and evaluate both our method and the Forward Backward (Eysenbach et al. 2018) method. For the Forward Backward method, we remove the training of the reach task by scripting it, making the learning problem easier. Additionally, we script the reach subtask, combine grasp and flipup into one forward task with horizon $T = 100$ (50 for each task) and 600 milestone images, and train insertion as the backward task. Even when making the learning problem easier for the Forward Backward method setting by scripting the reach subtask, our method outperforms the Forward Backward method. The discrepancies in the performance of the algorithms demonstrate the improvements in learning capacity as the granularity of the division of tasks increases.

For the Forward Backward method, we combine grasp, flipup, and hook into one forward task with horizon $T = 200$ (50 for grasp and flipup, 100 for hook) and 900 milestone images, and train unhook as the backward task. Once again, since our method outperforms the Forward Backward method, the results highlight the improvements in learning capacity as the granularity of the division of tasks increases.

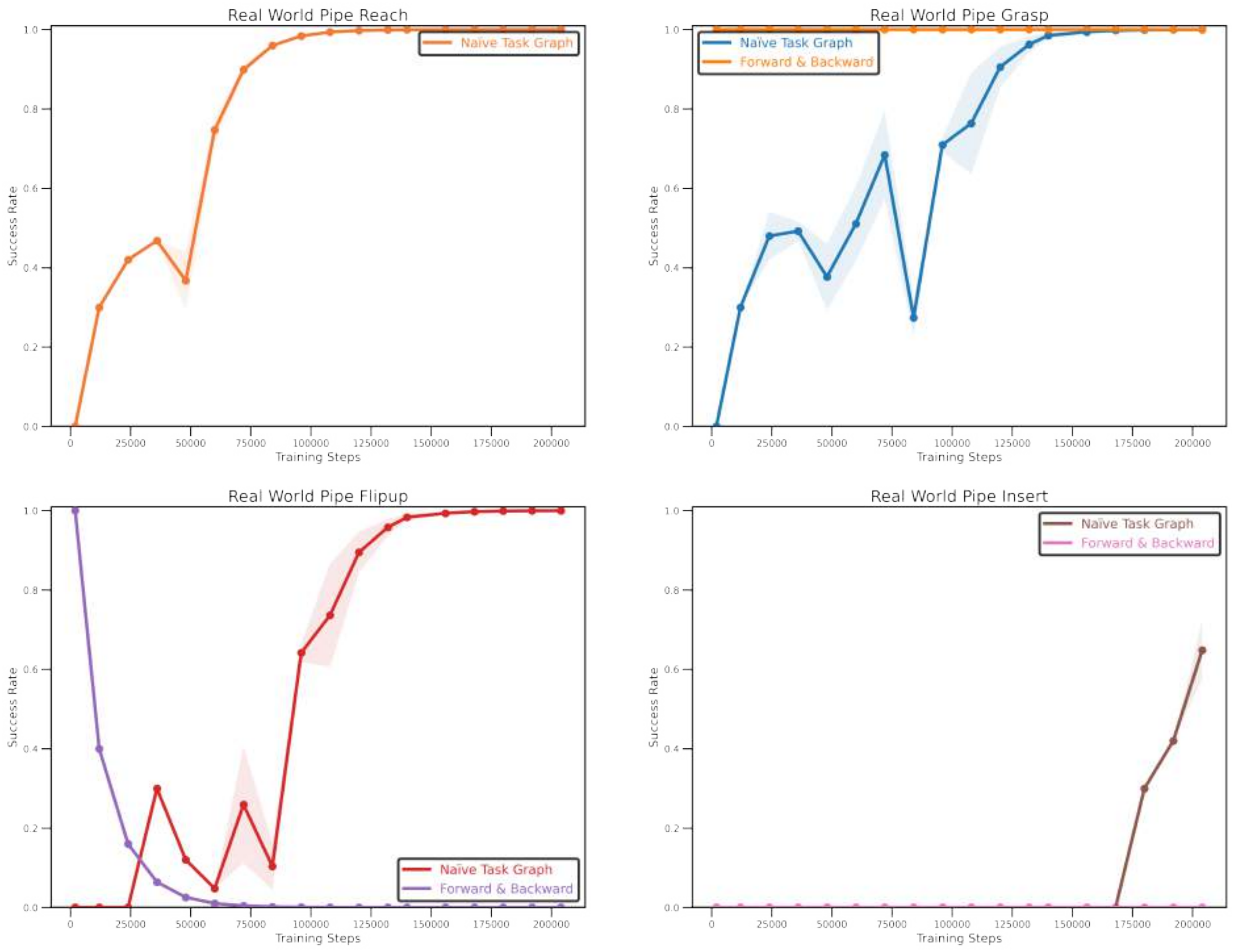


Figure 50: Success rate of each subtask on our real world insertion task. The reach task curve with Forward Backward method is omitted as mentioned above. The Forward Backward method is unable to learn the flipup and insertion task while our method with the naïve task graph achieves substantial learning progress across all tasks.

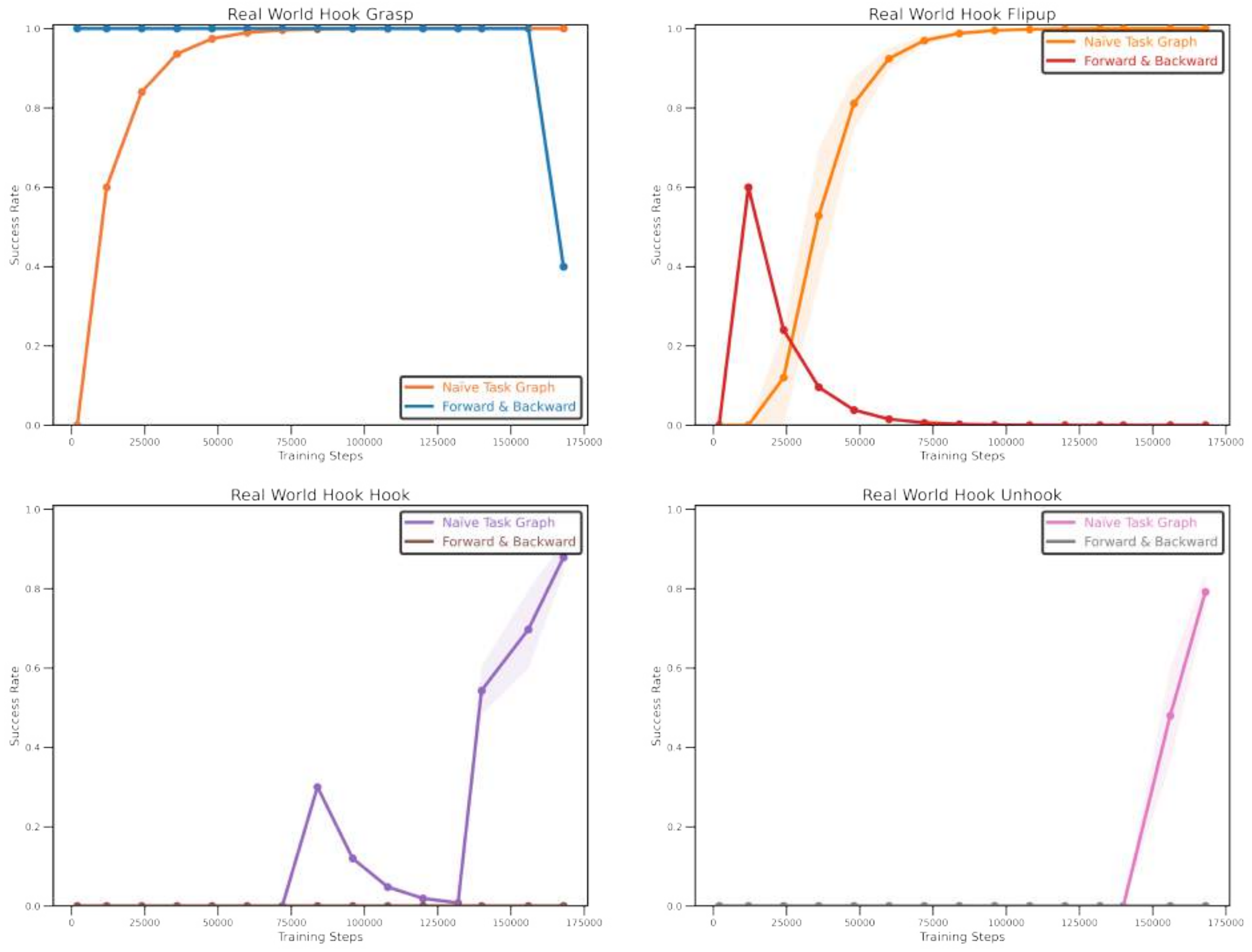


Figure 51: Success rate of each task on our real world hooking task. The Forward Backward method is unable to learn the flipup, hook, and unhooking subtasks while our method with the naïve task graph achieves substantial learning progress across all tasks.

E.4.2 *Comparison with Learned Task Graph*

Here we show the success rate of our approach using a learned task graph on our real world insertion task. Different then our simulated domain, we find that on our more challenging real world tasks, we obtain significantly faster convergence (approximately 150,000 steps vs. 200,000 steps) in terms of final insertion performance using our learned task graph.

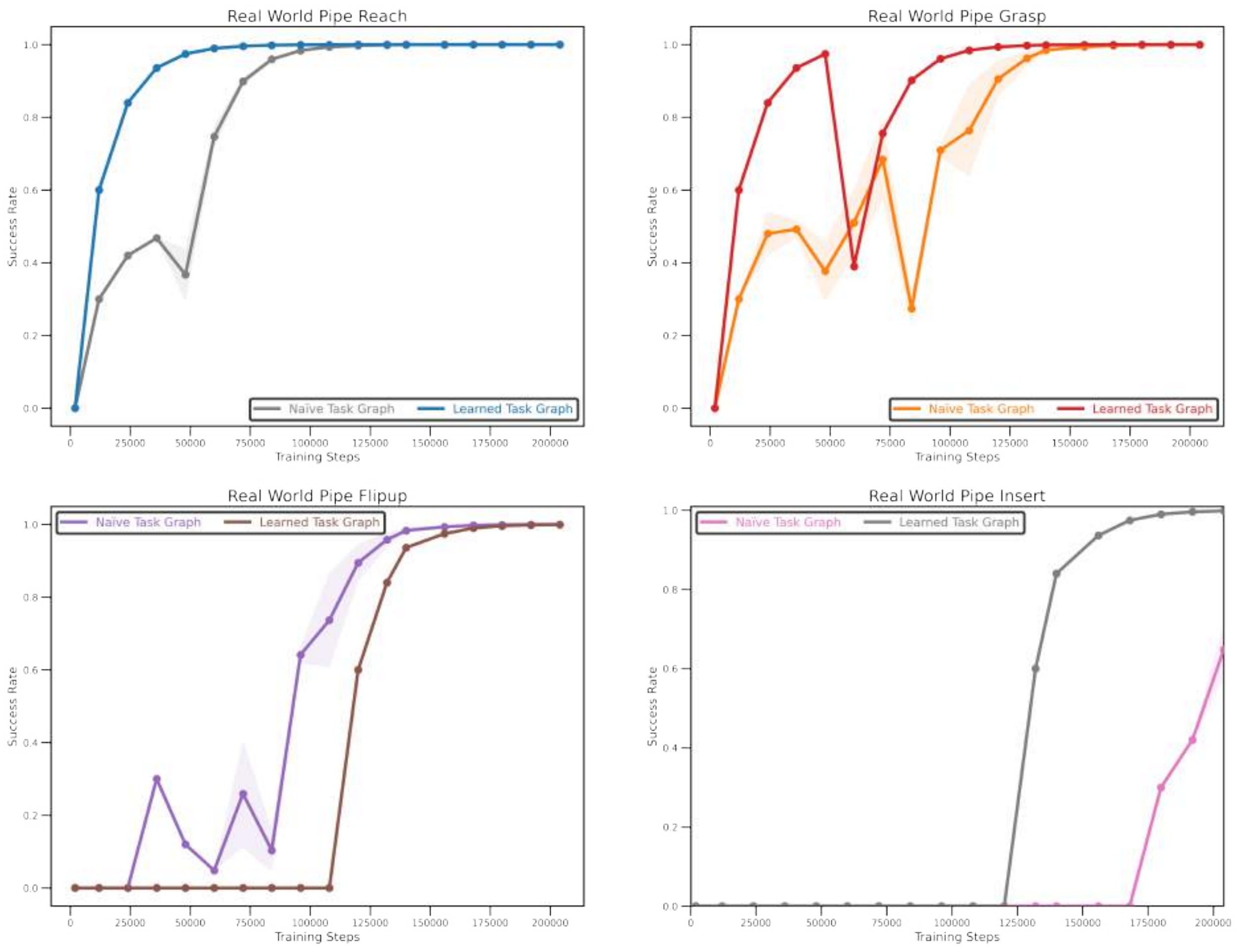


Figure 52: Success rate of each task on our real world insertion task. We find that using a learned task graph results in faster convergence on our real world robotic task, where the robot begins to consistently perform the task around 25% faster than the naïve task graph.

E.4.3 *Example Simulated Milestones*

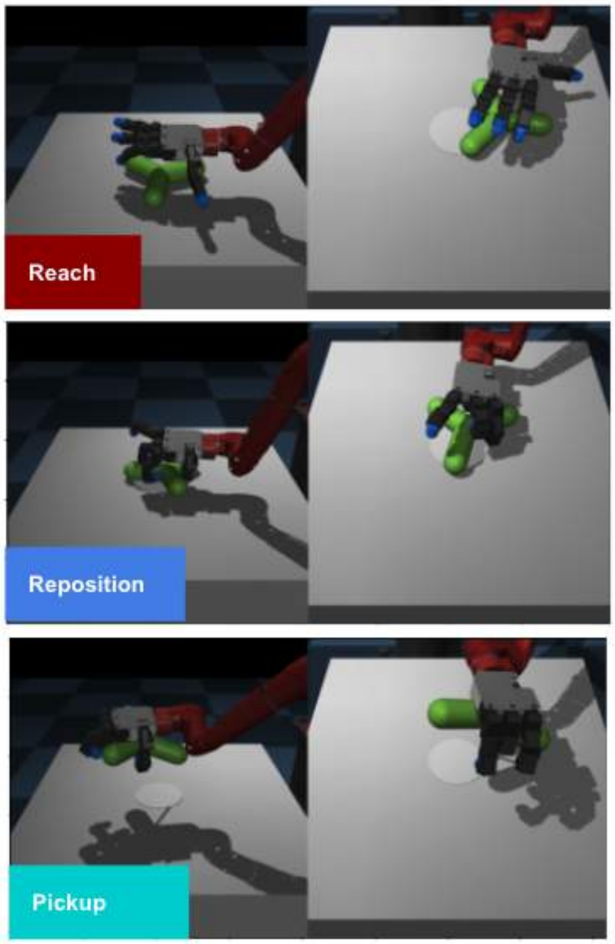


Figure 53: The experimental domain `DHandValve-v0` we study in this work. We consider a task where the simulated robot hand is required to pick up a three-pronged object. Our observations consist of images from two viewpoints (shown above), in addition to the robot's proprioceptive state. We assume no access to a ground truth reward function, nor to episodic resets. The labels in the bottom left corners were overlayed for visualization purposes.

F

APPENDIX: BENCHMARKING ALGORITHMS FOR AUTONOMOUS REINFORCEMENT LEARNING

F.0.1 *Goal-conditioned ARL*

This framework can be readily extended to goal-conditioned reinforcement learning ("Learning to achieve goals"; Schaul et al., 2015), which is also an area of study covered in some prior works on reset-free reinforcement learning (Sharma et al., 2021a). Assuming a goal space $\mathcal{G}$ and task-distribution $p_g : \mathcal{G} \mapsto \mathbb{R}_{\geqslant 0}$, assume that the algorithm $\mathbb{A} : \{s_i, a_i, s_{i+1}\}_{i=0}^{t-1} \mapsto (a_t, \pi_t)$, where $a_t \in \mathcal{A}$ and $\pi_t : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \mapsto \mathbb{R}_{\geqslant 0}$. Equation 19 can be redefined as follows:

$$J_D(\pi) = \mathbb{E}_{g \sim p_g, s_0 \sim \rho, a_t \sim \pi(\cdot|s_t, g), s_{t+1} \sim p(\cdot|s_t, a_t)} \Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, g) \Big] \tag{28}$$

Additionally, we will assume that the algorithm has access to a set of samples $g_i \sim p(g)$ from the goal distribution. The definitions for deployed policy evaluation and continuing policy evaluation remains the same.

F.0.2 *Reversibility and Non-Ergodic MDPS*

We expand on the discussion of reversibility in 7.3.2. We also discuss how we can deal with non-ergodic MDPs by augmenting the action space in MDP $\mathcal{M}_T$ with calls for extrinsic interventions.

Definition 4 (*Ergodic MDPs*)**.** *A MDP is considered ergodic if for all states* $s_a, s_b \in \mathcal{S}, \exists \pi$ *such that* $\pi(s_b) > 0$, *where* $\pi(s) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t p(s_t = s \mid s_0 = s_a, \pi)$ *denotes dis-*

counted state distribution induced by the policy π starting from the state $s_0 = s_a$ (Moldovan and Abbeel, 2012). A policy that assigns a non-zero probability to all actions on its support $\mathcal{A}$ ensures that all states in $\mathcal{S}$ are visited in the limit for ergodic MDPs, satisfying the condition above.

We adapt the ARL framework to develop learning algorithms for non-ergodic MDPs. In particular, we introduce a mechanism below for the agent to call for an intervention in the MDP $\mathcal{M}_T$. Our goal here is to show that our described ARL framework is general enough to include cases where a human is asked for help in irreversible states. Consider an augmented state space $\mathcal{S}^+ = \mathcal{S} \cup \mathbb{R}_{\geqslant 0}$ and an augmented action space $\mathcal{A}^+ = \mathcal{A} \cup \mathcal{A}_h$, where $\mathcal{A}_h$ denotes the action of asking for help via human intervention. A state $s^+ \in \mathcal{S}^+$ can be written as a state $s \in \mathcal{S}$, and $h \in \mathbb{R}_{\geqslant 0}$ which denotes the remaining budget for interventions into the training. The budget h is initialized to h_{max} when the training begins. The intervention can be requested by the agent itself using an action $a \in \mathcal{A}_h$ or it can enforced by the environment (for example, if the agent reaches certain detectable irreversible states in the environment and requests a human to bring it back into the reversible set of states). For an action $a \in \mathcal{A}_h$, the environment transitions from $(s, h) \to (s', h - c(s, a))$, where the next state s' depends on the nature of the requested intervention a and $c(s, a) : \mathcal{S} \times \mathcal{A}_h \mapsto \mathbb{R}_{\geqslant 0}$ denotes the cost of intervention. A similar transition occurs when the environment enforces the human intervention. When the cost exceeds the remaining budget, that is $h - c(s, a) \leqslant 0$, the environment transitions into an absorbing state $s_\emptyset$, and the training is terminated. We retain the definitions introduced in Sec 7.3.1.1 and 7.3.1.2 for evaluation protocols.[1] In this way, we see that we can actually encompass the setting of non-ergodic MDPs as well, with some reparameterization of the MDP.

F.0.3 *Relating Prior Works to ARL*

In this section, we connect the settings in prior works to our proposed autonomous reinforcement learning formalism. First, consider the typical reinforcement learning approach: Algorithm $\mathbb{A}$ assigns the rewards to transitions using $r(s, a)$, learns a policy π and outputs $\pi_t = \pi$ and $a_t \sim \pi$ (possibly adding noise to the action). The algorithm exclusively optimizes the reward function r throughout training.

Reconsider the door closing example: the agent needs to practice closing the door repeatedly, which requires opening the door repeatedly. However, if we optimize π to

1 The action space $\mathcal{A}_h$ is not available in $\mathcal{M}$, only in $\mathcal{M}_T$. This discrepancy should be considered when parameterizing π.

maximize r over its lifetime, it will never be incentivized to open the door to practice closing it again. In theory, assuming an ergodic MDP and that the exploratory actions have support over all actions, the agent will open the door given enough time, and the agent will practice closing it again. However, in practice, this can be quite an inefficient strategy to rely on and thus, prior reset-free reinforcement learning algorithms consider other strategies for exploration. To understand current work, we introduce the notion of a surrogate reward function $\tilde{r}_t : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. At every time step t, $\mathbb{A}$ outputs $a_t \sim \pi_e$ and π_t for evaluation, where π_e optimizes $\tilde{r}_t$ over the transitions seen till time $t-1$ [2]. Prior works on reset-free reinforcement learning can be encapsulated within this framework as different choices for surrogate reward function $\tilde{r}_t$. Some pertinent examples: Assuming r_ρ is some reward function designed that shifts the agent's state distribution towards initial state distribution ρ, alternating $\tilde{r}_t = r$ and $\tilde{r}_t = r_\rho$ for a fixed number of environment steps recovers the forward-backward reinforcement learning algorithms proposed by Han et al. (2015); Eysenbach et al. (2017). Similarly, R3L (Zhu et al., 2020a) can be understood as alternating between a perturbation controller optimizing a state novelty reward and the forward controller optimizing the task reward r. Recent work on using multi-task learning for reset-free reinforcement learning (Gupta et al., 2021b) can be understood as choosing $\tilde{r}_t(s_t, a_t) = \sum_{k=1}^{K} r_k(s_t, a_t)\mathbb{I}[s_t \in \mathcal{S}_k]$ such that $\mathcal{S}_1, \ldots \mathcal{S}_K$ is a partition of the state space $\mathcal{S}$ and only reward function r_k is active in the subset $\mathcal{S}_k$. Assuming the goal-conditioned autonomous reinforcement learning framework, the recently proposed algorithm VaPRL (Sharma et al., 2021a) can be understood as creating a curriculum of goals g_t such that at every step, the action $a_t \sim \pi(\cdot \mid s_t, g_t)$. The curriculum simplifies the task for the agent, bootstrapping on the success of easier tasks to efficiently improve $J_D(\pi)$.

F.0.4 *Environment Descriptions and Reward Functions*

Tabletop-Organization. The Tabletop-Organization task is a diagnostics object manipulation task proposed by Sharma et al., 2021a. The observation space is a 12 dimensional vector consisting of the object position, gripper position, gripper state, and the current goal. The action space is 3-D action that consists of a 2-D position delta and an automated gripper that can attach to the mug if the gripper is close enough to the object. The reward function is a sparse indicator function, $r(s, g) = \mathbb{1}(\|s - g\|_2 \leqslant 0.2)$. The agent is provided with 12 forward and 12 backward demonstrations, 3 for each of the 4 goal

2 This can also be captured in a multi-task reinforcement learning framework, where π_e, π_t are the same policy with different task variables as input.

positions.

Sawyer-Door. The Sawyer Door environment consists of a Sawyer robot with a 12 dimension observation space consisting of 3-D end effector position, 3-D door position, gripper state and desired goal. The action space is a 4-D action space that consisting of a 3-D end effector control and normalized gripper torque. Let s_d be dimensions of the observation corresponding to the door state, and g be the corresponding goal. The reward function is a sparse indicator function $r(s, g) = \mathbb{1}(\|s_d - g\|_2 \leqslant 0.08)$. The agent is provided with a 5 forward and 5 backward demos.

Sawyer-Peg. The Sawyer-Peg environment shares observation and action space as the Sawyer-Door environment. Let s_p be the state of the peg and g be the corresponding goal. The reward function is a sparse indicator function $r(s, g) = \mathbb{1}(\|s - g\|_2 \leqslant 0.05)$. The agent is provided with 10 forward and 10 backward demonstrations for this task.

Franka-Kitchen. The Franka-Kitchen environment consists of a 9-DoF Franka robot with an array of objects (microwave, two distinct burner, door) represented by a 14 dimensional vector. The reward function is composed of a dense reward that is a sum of the euclidean distance between the goal position of the arm and the current state plus shaped reward per object as described in Gupta et al., 2019. No demonstrations are provided for this task.

DHand-LightBulb. The DHand is a 4 fingered robot (16-DoF) mounted on a 6-DoF Sawyer Arm. The observation space of the DHand consists of a 30 dimensional observation and corresponding goal state. The observation is composed of a 16 dimensional hand position, 7 dimensional arm position, 3 dimension object position, 3 dimensional euler angle and a 6 dimensional vector representing the dimensional wise distance to between objects in the environment. The action space is a position delta over the combined 22 DoF of the robot. No demonstrations are provided for these tasks.

Minitaur-Pen. The Minitaur pen's observation space is the joint positions of its 8 links, their corresponding velocity, current torque, quaternion of its base position, and goal location in the pen. The action space is a 8 dimensional PD target. Let s_b be the 2-D position of the agent, and g be the corresponding goal. Let s_t be the current torques on the agent, and s_v be their velocities The reward for the agent is a dense reward $r(s, a) = -2.0 \cdot \|s_b - g\| + 0.02 \cdot \|s_v \cdot s_t\|$. No demonstrations are provided for these tasks.

F.0.5 *Algorithms*

We use soft actor-critic (Haarnoja et al., 2018d) as the base algorithm for our experiments in this paper. When available, the demonstrations are added to the replay buffer at the beginning of training. Further details on the parameters of the environments and algorithms are reported in Tables 11. The implementations will be open-sourced and implementation details can be found there.

Hyperparameter	**Value**
Actor-critic architecture	fully connected(256, 256)
Nonlinearity	ReLU
RND architecture	fully connected(256, 256, 512)
RND Gamma	0.99
Optimizer	Adam
Learning rate	3e-4
γ	0.99
Target update τ	0.005
Target update period	1
Batch size	256
Classifier batch size	128
Initial collect steps	10^3
Collect steps per iteration	1
Reward scale	1
Min log std	-20
Max log std	2

Table 11: Shared algorithm parameters.

Environment	**Training Horizon (H_T)**	**Evaluation Horizon (H_E)**	**Replay Buffer Capacity**
Tabletop-Organization	200,000	200	20,000,000
Sawyer-Door	200,000	300	20,000,000
Sawyer-Peg	100,000	200	20,000,000
Franka-Kitchen	100,000	400	10,000,000
DHand-Lightbulb	400,000	400	10,000,000
Minitaur-Pen	100,000	1000	10,000,000

Table 12: Environment specific parameters, including the training horizon (i.e. how frequently an intervention is provided), the evaluation horizon, and the replay buffer capacity.

F.0.6 *Evaluation Curves*

In this section, we plot deployed policy evaluation and continuing policy evaluation curves for different algorithms and different environments:

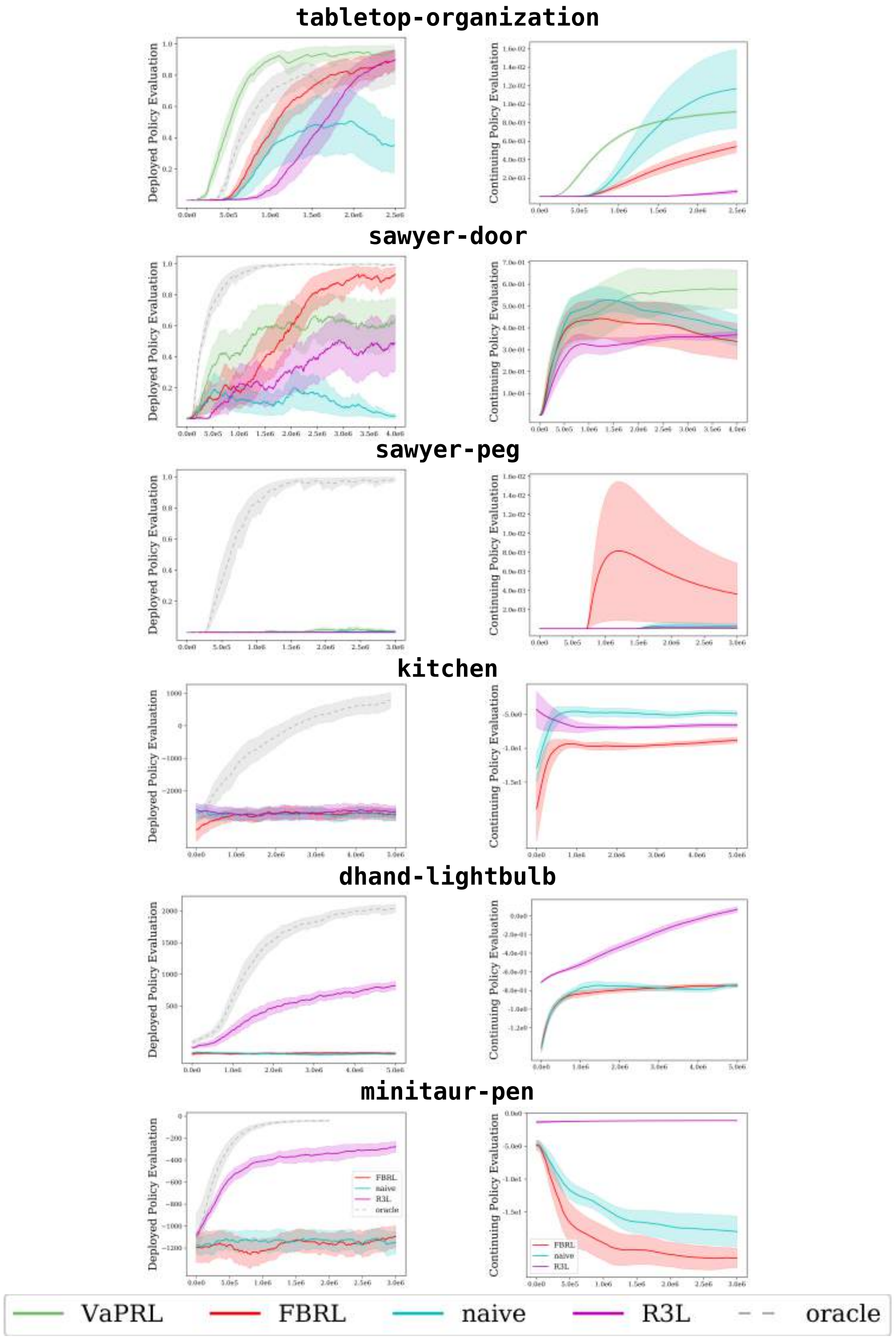


Figure 54: Deployed Policy Evaluation and Continuing Policy Evaluation per environment. Results and averaged over 5 seeds. Shaded regions denote 95% confidence bounds.