Learning and Analyzing Representations for Meta-Learning and Control

Kate Rakelly

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2020-224
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-224.html

December 18, 2020

Learning and Analyzing Representations for Meta-Learning and Control

by

Kate Rakelly

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Assistant Professor Sergey Levine, Chair
Professor Pieter Abbeel
Professor Alison Gopnik

Fall 2020

Learning and Analyzing Representations for Meta-Learning and Control

Abstract

Learning and Analyzing Representations for Meta-Learning and Control

by

Kate Rakelly

Doctor of Philosophy in Electrical Engineering

University of California, Berkeley

Assistant Professor Sergey Levine, Chair

While artificial learning agents have demonstrated impressive capabilities, these successes are typically realized in narrowly defined problems and require large amounts of labeled data. Our agents struggle to leverage what they already know to generalize to new inputs and acquire new skills quickly, abilities quite natural to humans. To learn and leverage the structure present in the world, we study data-driven abstractions of states and tasks. We begin with unsupervised state representation learning, in which the goal is to learn a compact state representation that discards irrelevant information but preserves the information needed to learn the optimal policy. Surprisingly, we find that several commonly used objectives are not guaranteed to produce sufficient representations, and demonstrate that our theoretical findings are are reflected empirically in simple visual RL domains.

Next, we turn to learning abstractions of tasks, a problem typically studied as meta-learning. Meta-learning is an approach to endow artificial agents with this capability that leverages a set of related training tasks to learn an adaptation mechanism that can be used to acquire new skills from little supervision. We adopt an inference perspective that casts meta-learning as learning probabilistic task representations, framing the problem of learning to learn as learning to infer hidden task variables from experience. Leveraging this viewpoint, we propose meta-learning algorithms for diverse applications: image segmentation, state-based robotic control, and robotic control from sensory observations. We find that an inference approach to these problems constitutes an efficient and practical choice, while also revealing deeper connections between meta-learning and other concepts in statistical learning.

ACKNOWLEDGMENTS

Thank you to Sergey Levine for your guidance as my advisor over the past four years. Your intellect, dedication, and excitement continue to inspire me. I would also like to thank Alyosha Efros and Trevor Darrell who advised me during the first few years, as well as Pieter Abbeel and Alison Gopnik for serving on the qualifying exam and dissertation committees.

I would like to especially thank those I worked particularly closely with on the research in this thesis: Evan Shelhamer, Aurick Zhou, Anusha Nagabandi, and Tony Zhao. I enjoyed our whiteboard scheming, late night takeout dinners, and of course celebratory tequila sipping after paper submissions! Special thanks to Aurick and Tony for tolerating my obsession with making the perfect plot. Thanks to Evan for your research mentorship during my first two years, as well as the many squares of Dandelion chocolate and cups of tea. Your high standards for every aspect of research were formative for me.

Thank you to Abhishek Gupta for your friendship and excellent advice on everything from troubleshooting MuJoCo to how to break down a research problem. Thank you to Rowan McAllister for being a fantastic person to sit next to. Your optimism and excitement for your work are infectious!

Prior to my PhD, I was convinced to apply to graduate school thanks to experiences I had working with Shiry Ginosar, Alyosha Efros, Bryan Russell, Claire Tomlin, and Insoon Yang. Floraine Berthuozoz also supported me in undergraduate research and encouraged me to pursue a PhD. I discovered machine learning thanks to my dear friend Huda Khayrallah who convinced me to take AI classes with her. Thank you to my parents Lee and Tom for your continued support during these last five years, and for passing on to me your love for math and science. Yes, I'm finally done with school!

I was fortunate to do my PhD in the beautiful, diverse, and funky city of Berkeley. After nearly ten years living here, I still find new things to discover while enjoying old favorites, particularly Indian Rock Park where I have spent many happy hours bouldering and watching sunsets. Of course what makes the city is the people who live here - thank you to all the friends who have made the last five years of my life so memorable. To Juniele Rabelo de Almeide, Ignasi Clavera, Kate Beck, Maria Eckstein, and Carlos Florensa, thank you for your unwavering support and encouraging me to keep going!

CONTENTS

1

INTRODUCTION

To operate fully autonomously in the world, artificial agents must be able to handle the continuous variations in their environment and in the tasks they must perform. For example, a general-purpose household robot must contend with thousands of different objects in ever-changing configurations, as well as varying tasks to complete. Humans possess the ability to quickly adapt to unforeseen circumstances by generalizing existing concepts when appropriate and acquiring new concepts from few examples. How can we endow artificial agents with these same abilities?

The standard machine learning approach of recent years to solve any particular task is to collect a large amount of labeled data and train a randomly initialized deep network end-to-end. While important inductive biases exist in the network architecture, this approach largely forgoes built-in knowledge in favor of data driven learning. This approach has been widely and wildly successful for applications ranging from scene understanding (Krizhevsky et al., 2012; Donahue et al., 2015; Shelhamer et al., 2016a) to machine translation (Hassan et al., 2018), playing video games (Vinyals et al., 2019), and object grasping with robotic arms (Kalashnikov et al., 2018a), to name a few. However, from the perspective of natural intelligence, and from the desire for data efficiency, it makes little sense to start "tabula rasa" for each new skill an agent learns. Most of the tasks we wish an agent to perform share overlapping structure, whether it's a concept of objects, robustness to lighting conditions, or control of a robot's limbs. Learning and representing this shared structure would allow agents to quickly acquire new skills by leveraging previous knowledge.

Structure exists across percepts, in the environment dynamics, and in the reward functions for different tasks. *Abstractions* capture this important structure while discarding irrelevant information. For example, consider a robot cooking in the kitchen. Abstractions of observations might ignore irrelevant information such as the lighting of the room, the color of the wall, or the noise in the robot's sensors. This problem can be tack-

led as latent state estimation – from the history of observations, the agent estimates its current state.

We begin our study of learned abstractions with unsupervised latent state estimation. These methods learn to estimate relevant information without any supervisory signals. An appealing and popular way to achieve this is by maximizing mutual information between variables (Watter et al., 2015; Pathak et al., 2017; Oord et al., 2018). While prior work has largely focused on how the mutual information can be estimated in high dimensions, we seek instead to understand which objectives are sufficient for learning and representing the optimal policy. In Chapter 2, we analyze three objectives proposed in prior work and find that in fact two are not guaranteed to yield sufficient representations. Further, we find that the sufficiency of a representation can indeed affect the performance of the learned policy in image-based deep RL.

The similarity in underlying states between different percepts is not the only structure we can leverage. There is also structure in agent behavior. For example, cooking tasks like stirring, crushing, and chopping involve manipulating the robot's end effector to use a tool in a repetitive motion. We would like to capture this structure, so that the agent can leverage it to learn new skills more quickly. This problem is often viewed as the problem of meta-learning, or learning to learn.

Meta-learning approaches optimize for effective transfer to enable the fast learning of new concepts (Schmidhuber, 1987; Bengio et al., 1990; Thrun and Pratt, 1998). In the context of modern deep learning approaches, meta-learning algorithms typically assume a *meta-training* phase in which the algorithm has access to a set of related tasks, which are used to learn an adaptation mechanism for quickly learning new tasks. The adaptation mechanism may be learned from data during the course of meta-training, or defined beforehand. Meta-training directly optimizes for the performance of the adapted model across the set of meta-training tasks with shared structure. Meta-learning algorithms are often viewed as learning a learning algorithm (Wang and Hebert, 2016; Duan et al., 2016; Santoro et al., 2016; Ravi and Larochelle, 2017), or more recently as learning a weight initialization for adaptation via gradient descent (Finn et al., 2017a; Finn and Levine, 2017). Notably, these algorithms have little in common with the state representation learning methods discussed above.

In this thesis, we argue that meta-learning and state representation learning are instantiations of a more general principle for learning abstractions. State representation learning algorithms learn abstractions of the current state of the agent, meta-learning algorithms learn abstractions of what the agent should *do*. Drawing on this connection, we advocate a representation learning approach to meta-learning. We frame meta-learning

as learning to infer representations of *tasks*, by performing probabilistic inference over hidden task variables. Our perspective is related to hierarchical Bayesian approaches to meta-learning, which have proven useful for few-shot learning (Tenenbaum, 1999; Lawrence and Platt, 2004; Fei-Fei et al., 2006; Daumé III, 2009; B. Lake et al., 2011; Edwards and Storkey, 2016). We demonstrate that this perspective leads to efficient and elegant algorithms for diverse applications including image segmentation and robotic control from images. In fact, we will show in Chapter 5 that the exact same algorithm can perform both state estimation and meta-learning; the same latent state can capture where the agent is and what the agent should do.

The main contributions of this thesis are the following:

- In Chapter 2, we focus on the state representation learning problem and evaluate the sufficiency for control of representations resulting from several popular mutual information-based learning objectives.
- In Chapter 3 we introduce and formalize the meta-learning framework for supervised learning and reinforcement learning.
- In Chapter 4, we propose a task-inference based meta-learning algorithm for guiding segmentation from point-wise annotations. This work was presented at ICLR 2018, Workshop Track.
- In Chapter 5, we explore the idea of meta-RL as probabilistic inference and develop two meta-RL algorithms. The first, PEARL, performs off-policy model-free meta-RL by inferring a belief over a latent task variable via variational inference. This work was published at the International Conference of Machine Learning (ICML) 2019. The second, MELD, re-purposes latent state models to perform joint state and task inference. This work was published in the Conference on Robot Learning (CoRL) 2020.
- Finally, in Chapter 6, we conclude with future directions for learning data-driven abstractions that capture meaningful structure.

2

WHAT MUTUAL INFORMATION-BASED REPRESENTATION LEARNING OBJECTIVES ARE SUFFICIENT FOR CONTROL?

While deep reinforcement learning (RL) algorithms are capable of learning policies from high-dimensional observations, such as images (Mnih et al., 2013; A. Lee et al., 2019; Kalashnikov et al., 2018b), in practice policy learning faces a bottleneck in acquiring useful representations of the observation space (Shelhamer et al., 2016b). State representation learning approaches aim to remedy this issue by learning structured and compact representations on which to perform RL. While a wide range of representation learning objectives have been proposed (Lesort et al., 2018), a particularly appealing class of methods that is amenable to rigorous analysis is based on maximizing mutual information (MI) between variables. In the unsupervised learning setting, this is often realized as the InfoMax principle (Linsker, 1988; Bell and Sejnowski, 1995), which maximizes the mutual information between the input and its latent representation subject to domain-specific constraints. This approach has been widely applied in unsupervised learning in the domains of image, audio, and natural language understanding (Oord et al., 2018; Hjelm et al., 2018; Ravanelli and Bengio, 2019). In RL, the variables of interest for MI maximization are sequential states, actions, and rewards (see Figure 1). As we will discuss, several popular methods for representation learning in RL involve mutual information maximization with different combinations of these variables (Anand et al., 2019; Oord et al., 2018; Pathak et al., 2017; Shelhamer et al., 2016b).

A useful representation should retain the factors of variation that are necessary to learn and represent the optimal policy or the optimal value function, and discard irrelevant and redundant information. While much prior work has focused on the problem of how to optimize various mutual information objectives in high dimensions (J. Song and Ermon, 2019; Belghazi et al., 2018; Oord et al., 2018; Hjelm et al., 2018), we focus instead on whether the representations that maximize these objectives are actually theoretically sufficient for learning and representing the optimal policy or value function.

We find that some commonly used objectives are insufficient given relatively mild and common assumptions on the structure of the MDP, and identify other objectives which are sufficient. We show these results theoretically and illustrate the analysis empirically in didactic examples in which MI can be computed exactly. Our results provide some guidance to the deep RL practitioner on when and why objectives may be expected to work well or fail, and also provide a framework to analyze newly proposed representation learning objectives based on MI. To investigate how our theoretical results pertain to deep RL, we compare the performance of RL agents in a simulated game trained with state representations learned by maximizing the MI objective given visual inputs. The experimental results corroborate our theoretical findings, and demonstrate that the sufficiency of a representation can have a substantial impact on the performance of an RL agent that uses that representation.

2.1 RELATED WORK

In this chapter, we analyze several widely used mutual information objectives for control. In this section we first review MI-based unsupervised learning, then the application of these techniques to the RL setting. Finally, we discuss alternative perspectives on representation learning in RL.

Mutual information-based unsupervised learning. Mutual information-based methods are particularly appealing for representation learning as they admit both rigorous analysis and intuitive interpretation. Tracing its roots to the InfoMax principle (Linsker, 1988; Bell and Sejnowski, 1995), a common technique is to maximize the MI between the input and its latent representation subject to domain-specific constraints (Becker and Hinton, 1992). This technique has been applied to learn representations for natural language (Devlin et al., 2019), video (Sun et al., 2019), and images (Bachman et al., 2019; Hjelm et al., 2018). A major challenge to using MI maximization methods in practice is the difficulty of estimating MI from samples (McAllester and Statos, 2018) and with high-dimensional inputs (J. Song and Ermon, 2019). Much recent work has focused on improving MI estimation via variational methods (J. Song and Ermon, 2019; Poole et al., 2019; Oord et al., 2018; Belghazi et al., 2018). In this work we are concerned with analyzing the MI objectives, and not the estimation method. In our experiments with image observations, we make use of noise contrastive estimation methods (Gutmann and Hyvärinen, 2010), though other choices could also suffice.

Mutual information objectives in RL. Reinforcement learning adds aspects of temporal structure and control to the standard unsupervised learning problem discussed

above (see Figure 1). This structure can be leveraged by maximizing MI between sequential states, actions, or combinations thereof. Some works omit the action, maximizing the MI between current and future states (Anand et al., 2019; Oord et al., 2018; Stooke et al., 2020). Much prior work learns latent forward dynamics models (Watter et al., 2015; Karl et al., 2016; M. Zhang et al., 2018; Hafner et al., 2019; A. Lee et al., 2019), related to the forward information objective we introduce in Section 2.3. Multi-step inverse models, closely related to the inverse information objective (Section 2.3), have been used to learn control-centric representations (Yu et al., 2019b; Gregor et al., 2016). Single-step inverse models have been deployed as regularization of forward models (A. Zhang et al., 2018; Agrawal et al., 2016) and as an auxiliary loss for policy gradient RL Shelhamer et al., 2016b; Pathak et al., 2017. The MI objectives that we study have also been used as reward bonuses to improve exploration, without impacting the representation, in the form of empowerment (Klyubin et al., 2008; Klyubin et al., 2005; Mohamed and Rezende, 2015; Leibfried et al., 2019) and information-theoretic curiosity (Still and Precup, 2012).

Representation learning for reinforcement learning. In RL, the problem of finding a compact state space has been studied as state aggregation or abstraction (Bean et al., 1987; L. Li et al., 2006). Abstraction schemes include bisimulation (Givan et al., 2003), homomorphism (Ravindran and Barto, 2003), utile distinction (McCallum, 1996), and policy irrelevance (Jong and Stone, 2005). While efficient algorithms exist for MDPs with known transition models for some abstraction schemes such as bisimulation (Ferns et al., 2006; Givan et al., 2003), in general obtaining error-free abstractions is highly impractical for most problems of interest. For approximate abstractions prior work has bounded the sub-optimality of the policy (Bertsekas, Castanon, et al., 1988; Dean and Givan, 1997; Abel et al., 2016) as well as the sample efficiency (Lattimore and Szepesvari, 2019; Van Roy and Dong, 2019; Du et al., 2019), with some results extending to the deep learning setting (Gelada et al., 2019; Nachum et al., 2018). In this chapter, we focus on whether a representation can be used to learn the optimal policy, and not the tractability of learning. Alternatively, priors based on the structure of the physical world can be used to guide representation learning (Jonschkowski and Brock, 2015). In deep RL, many auxiliary objectives distinct from the objectives that we study have been proposed, including meta-learning general value functions (Veeriah et al., 2019), predicting multiple value functions (Bellemare et al., 2019; Fedus et al., 2019; Jaderberg et al., 2016) and predicting domain-specific measurements (Mirowski, 2019; Dosovitskiy and Koltun, 2016). We restrict our analysis to objectives that can be expressed as MI-maximization.

2.2 REPRESENTATION LEARNING FOR RL

The goal of representation learning for RL is to learn a compact representation of the state space that discards irrelevant and redundant information. In this section we formalize each part of this statement, starting with defining the RL problem and representation learning in the context of RL. We then propose and define the metric of sufficiency to evaluate the usefulness of a representation.

2.2.1 *Preliminaries*

We begin with brief preliminaries of reinforcement learning and mutual information.

REINFORCEMENT LEARNING. A Markov decision process (MDP) is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ the set of actions, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ the state transition distribution, and $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ the reward function. We will use capital letters to refer to random variables and lower case letters to refer to values of those variables (e.g., S is the random variable for the state and $\mathbf{s}$ is a specific state). Throughout our analysis we will often be interested in multiple reward functions, and denote a set of reward functions as $\mathcal{R}$. The objective of RL is to find a policy that maximizes the sum of discounted returns $\bar{R}$ for a given reward function r, and we denote this optimal policy as $\pi^*_r = \arg\max_\pi \mathbb{E}_\pi[\sum_t \gamma^t r(S_t, A_t)]$ for discount factor γ. We also define the optimal Q-function as $Q^*_r(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{\pi^*}[\sum_{t=1}^{\infty} \gamma^t r(S_t, A_t) | \mathbf{s}_t, \mathbf{a}_t]$. The optimal Q-function satisfies the recursive Bellman equation, $Q^*_r(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)} \arg\max_{\mathbf{a}_{t+1}} Q^*_r(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})$. The optimal policy and the optimal Q-function are related according to $\pi^*(\mathbf{s}) = \arg\max_{\mathbf{a}} Q^*(\mathbf{s}, \mathbf{a})$.

MUTUAL INFORMATION. In information theory, the mutual information (MI) between two random variables, X and Y, is defined as (Cover, 1999):

$$I(X; Y) = \mathbb{E}_{p(x,y)} \log \frac{p(x, y)}{p(x)p(y)} = H(X) - H(X|Y). \tag{1}$$

The first definition indicates that MI can be understood as a relative entropy (or KL-divergence), while the second underscores the intuitive notion that MI measures the reduction in the uncertainty of one random variable from observing the value of the other.

REPRESENTATION LEARNING FOR RL. The goal of representation learning for RL is to find a compact representation of the state space that discards details in the state irrelevant for representing the policy or value function, while preserving task-relevant information (see Figure 1). While state aggregation methods typically define deterministic rules to group states in the representation (Bean et al., 1987; L. Li et al., 2006), MI-based representation learning methods used for deep RL treat the representation as a random variable (Nachum et al., 2018; Oord et al., 2018; Pathak et al., 2017). Accordingly, we formalize a representation as a stochastic mapping between original state space and representation space.

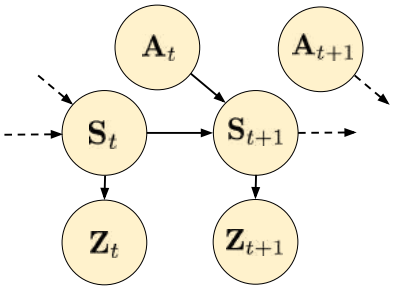


Figure 1: State representation learning: estimate representation Z from original state S.

Definition 1. *A **stochastic representation** $\phi_Z(\mathbf{s})$ is a mapping from states $\mathbf{s} \in \mathcal{S}$ to a probability distribution $p(Z|S = \mathbf{s})$ over elements of a new representation space $z \in \mathcal{Z}$.*

In this work we consider learning state representations from data by maximizing an objective $\mathbb{J}$. Given $\mathbb{J}$, we define the set of representations that maximize this objective as $\Phi_{\mathbb{J}} = \{\phi_Z\}$ s.t. $\phi_Z \in \arg\max \mathbb{J}(\phi)$. Unlike problem formulations for partially observed settings (Watter et al., 2015; Hafner et al., 2019; A. Lee et al., 2019), we assume that S is a Markovian state; therefore the representation for a given state is conditionally independent of the past states, a common assumption in the state aggregation literature (Bean et al., 1987; L. Li et al., 2006). See Figure 1 for a depiction of the graphical model.

2.2.2 *Sufficient Representations for RL*

We now turn to the problem of evaluating stochastic representations for RL. Intuitively, we expect a useful state representation to be capable of representing the optimal policy in the original state space.

Definition 2. *A representation ϕ_Z is π^***-sufficient** with respect to a set of reward functions $\mathcal{R}$ if $\forall r \in \mathcal{R}$, $\phi_Z(\mathbf{s}_1) = \phi_Z(\mathbf{s}_2) \implies \pi^*_r(A|\mathbf{s}_1) = \pi^*_r(A|\mathbf{s}_2)$.*

When a stochastic representation ϕ_Z produces the same distribution over the representation space for two different states $\mathbf{s}_1$ and $\mathbf{s}_2$ we say it *aliases* these states. Unfortunately, as already proven in Theorem 4 of L. Li et al. (2006) for the more restrictive case of deterministic representations, being able to represent the optimal policy does not guarantee that it can be learned via RL in the representation space. Accordingly, we define a stricter notion of sufficiency that *does* guarantee the convergence of Q-learning to the optimal policy in the original state space (refer to Theorem 4 of L. Li et al. (2006) for the proof of this).

Definition 3. *A representation ϕ_Z is Q^*-**sufficient** with respect to a set of reward functions $\mathcal{R}$ if $\forall r \in \mathcal{R}$, $\phi_Z(\mathbf{s}_1) = \phi_Z(\mathbf{s}_2) \implies \forall \mathbf{a}, Q^*_r(\mathbf{a}, \mathbf{s}_1) = Q^*_r(\mathbf{a}, \mathbf{s}_2)$.*

Note that Q^*-sufficiency implies π^*-sufficiency since the optimal policy and the optimal Q-function are directly related via $\pi^*_r(s) = \arg\max_a Q^*_r(s, a)$ (Sutton and Barto, 2018); however the converse is not true. We emphasize that while Q^*-sufficiency guarantees convergence, it does not guarantee tractability, which has been explored in prior work (Lattimore and Szepesvari, 2019; Du et al., 2019).

We will further say that an *objective* $\mathbb{J}$ is sufficient with respect to some set of reward functions $\mathcal{R}$ if all the representations that maximize that objective $\Phi_{\mathbb{J}}$ are sufficient with respect to every element of $\mathcal{R}$ according to the definition above. Surprisingly, we will demonstrate that not all commonly used objectives satisfy this basic qualification even when $\mathcal{R}$ contains a single known reward function.

2.3 Mutual information for representation learning in RL

In our study, we consider several MI objectives proposed in the literature.

Forward information: A commonly sought characteristic of a state representation is to ensure it retains maximum predictive power over future state representations. This property is satisfied by representations maximizing the following MI objective,

$$\mathbb{J}_{fwd} = I(Z_{t+k}; Z_t, A_t) = H(Z_{t+k}) - H(Z_{t+k}|Z_t, A_t). \tag{2}$$

We suggestively name this objective "forward information" due to the second term, which is the entropy of the forward dynamics distribution. This objective is related to that proposed in Nachum et al. (2018), where they consider a sequence of actions.

STATE-ONLY TRANSITION INFORMATION: Several popular methods (Oord et al., 2018; Anand et al., 2019; Stooke et al., 2020) optimize a similar objective, but do not include the action:

$$\mathbb{J}_{state} = I(Z_{t+k}; Z_t) = H(Z_{t+k}) - H(Z_{t+k}|Z_t). \tag{3}$$

As we will show, the exclusion of the action can have a profound effect on the characteristics of the resulting representations.

INVERSE INFORMATION: Another commonly sought characteristic of state representations is to retain maximum predictive power of the action distribution that could have generated an observed transition from $\mathbf{s}_t$ to $\mathbf{s}_{t+1}$. Such representations can be learned by maximizing the following information theoretic objective:

$$\mathbb{J}_{inv} = I(A_t; Z_{t+k}|Z_t) = H(A_t|Z_t) - H(A_t|Z_t, Z_{t+k}) \tag{4}$$

We suggestively name this objective "inverse information" due to the second term, which is the entropy of the inverse dynamics. A wide range of prior work learns representations by optimizing closely related objectives (Gregor et al., 2016; Shelhamer et al., 2016b; Agrawal et al., 2016; Pathak et al., 2017; Yu et al., 2019b; A. Zhang et al., 2018). Intuitively, inverse models allow the representation to capture only the elements of the state that are necessary to predict the action, allowing the discard of potentially irrelevant information.

S

2.4 SUFFICIENCY ANALYSIS

In this section we analyze the sufficiency for control of representations obtained by maximizing each objective presented in Section 2.3. To focus on the representation learning problem, we decouple it from RL by assuming access to a dataset of transitions collected with a policy that reaches all states with some probability, which can then be used to learn the desired representation. We also assume that distributions, such as the dynamics or inverse dynamics, can be modeled with arbitrary accuracy, and that the maximizing set of representations for a given objective can be computed. While these assumptions might be relaxed in any practical RL algorithm, and exploration plays a confounding role, studying these objectives under such simplifying assumptions allows us to compare them in terms of sufficiency on an equal playing field, isolating the role of representation learning from other confounding components of a complete RL algorithm.

2.4.1 *Forward Information*

In this section we show that a representation that maximizes $\mathbb{J}_{\text{fwd}}$ is sufficient for optimal control under any reward function. This result aligns with intuition that a representation that captures forward dynamics can represent everything predictable in the state space, and can thus be used to learn the optimal policy for any task. Note that this strength can also be a weakness if there are many predictable elements that are irrelevant for downstream tasks, since the representation retains more information than is needed for the task.

Proposition 1. $\mathbb{J}_{\text{fwd}}$ *is sufficient for all reward functions.*

Proof. (Sketch) We first show that if Z_t, A_t are maximally informative of Z_{t+k}, they are also maximally informative of the return $\bar{R}_t$. Due to the Markov structure, $\mathbb{E}_{p(Z_t|S_t=\mathbf{s})} p(\bar{R}_t|Z_t, A_t) = p(\bar{R}_t|S_t = \mathbf{s}, A_t)$. In other words, given $\phi_{\mathcal{Z}}$, additionally knowing S doesn't change our belief about the future return. The Q-value is the expectation of the return, so Z has as much information about the Q-value as S does. The full proof can be found in Appendix A.1. □

2.4.2 *State-only Transition Information*

While $\mathbb{J}_{\text{state}}$ is closely related to $\mathbb{J}_{\text{fwd}}$, we now show that it is not sufficient.

Proposition 2. $\mathbb{J}_{\text{state}}$ *is not sufficient for all reward functions.*

Proof. Consider the counter-example in Figure 2. Suppose that the two actions $\mathbf{a}_0$ and $\mathbf{a}_1$ are equally likely under the policy distribution. Each state gives no information about which of the two possible next states is more likely; this depends on the action. Therefore, a representation maximizing $\mathbb{J}_{\text{state}}$ is free to alias states with the same next-state distribution, such as $\mathbf{s}_0$ and $\mathbf{s}_3$. An alternative view is that such a representation can maximize $\mathbb{J}_{\text{state}} = H(Z_{t+k}) - H(Z_{t+k}|Z_t)$ by reducing both terms in equal amounts - aliasing $\mathbf{s}_0$ and $\mathbf{s}_3$ decreases the marginal entropy as well as the entropy of predicting the next state starting from $\mathbf{s}_1$ or $\mathbf{s}_2$. However, this aliased representation is not capable of representing the optimal policy which must distinguish $\mathbf{s}_0$ and $\mathbf{s}_3$ in order to choose the correct action to reach $\mathbf{s}_2$, which yields reward. □

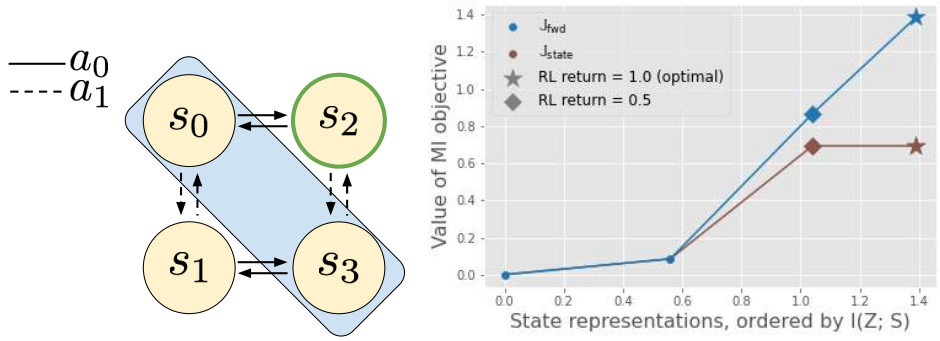


Figure 2: (left) A representation that aliases the states $\mathbf{s}_0$ and $\mathbf{s}_3$ into a single state maximizes $\mathbb{J}_{state}$ but is not sufficient to represent the optimal policy which must choose different actions in $\mathbf{s}_0$ and $\mathbf{s}_3$ to reach $\mathbf{s}_2$ which yields reward. (right) Values of $\mathbb{J}_{state}$ and $\mathbb{J}_{fwd}$ for a few representative state representations, ordered by increasing $I(Z;S)$. The representation that aliases $\mathbf{s}_0$ and $\mathbf{s}_3$ (plotted with a diamond) maximizes $\mathbb{J}_{state}$, but the policy learned with this representation may not be optimal (as shown here). The original state representation (plotted with a star) is sufficient.

2.4.3 *Inverse Information*

Here we show that representations that maximize $\mathbb{J}_{inv}$ are not sufficient for control in all MDPs. Intuitively, one way that the representation can be insufficient is by retaining only controllable state elements, while the reward function depends on state elements outside the agent's control. We then show that additionally representing the immediate reward is not enough to resolve this issue.

Proposition 3. *$\mathbb{J}_{inv}$ is not sufficient for all reward functions. Additionally, adding $I(R_t; Z_t)$ to the objective does not make it sufficient.*

Proof. Consider the MDP illustrated in Figure 3, and the representation that aliases the states $\mathbf{s}_0$ and $\mathbf{s}_1$. The same actions taken from these states lead to different next states which may have different rewards ($\mathbf{a}_0$ leads to the reward from $\mathbf{s}_0$ while $\mathbf{a}_1$ leads to the reward from $\mathbf{s}_1$). However, this representation maximizes $\mathbb{J}_{inv}$ because given each pair of states, the action is identifiable. Interestingly, this problem cannot be remedied by simply requiring that the representation also be capable of predicting immediate rewards. The same counterexample holds since we assumed $\mathbf{s}_0$ and $\mathbf{s}_1$ have the same reward. □

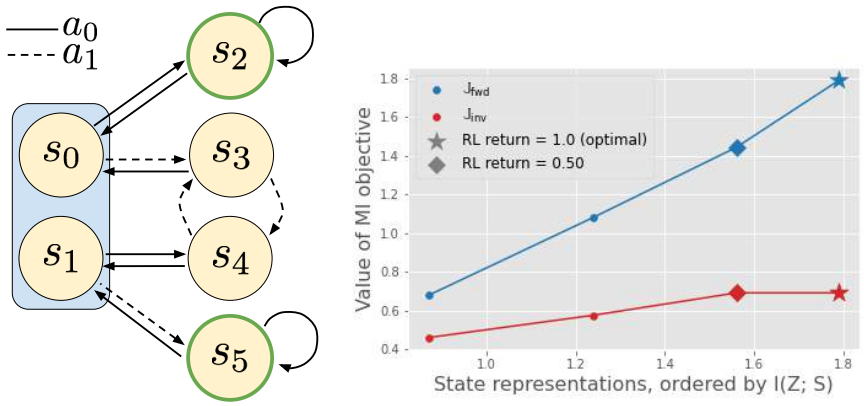


Figure 3: (left) In this MDP, a representation that aliases the states $\mathbf{s}_0$ and $\mathbf{s}_1$ into a single state maximizes $\mathbb{J}_{inv}$, yet is not sufficient to represent the optimal policy, which must distinguish between $\mathbf{s}_0$ and $\mathbf{s}_1$ in order to take a different action (towards the high-reward states outlined in green). (right) Values of $\mathbb{J}_{inv}$ and $\mathbb{J}_{fwd}$ for a few selected state representations, ordered by increasing $I(Z;S)$. The representation that aliases $\mathbf{s}_0$ and $\mathbf{s}_1$ (plotted with a diamond) maximizes $\mathbb{J}_{inv}$, but is not sufficient to learn the optimal policy. Note that this counterexample holds also for $\mathbb{J}_{inv} + I(R;Z)$.

2.5 EXPERIMENTS

In this section, we present experiments studying MI-based representation learning with image observations, to analyze whether the conclusions of our theoretical analysis hold in practice. Our goal is not to show that any particular method is necessarily better or worse, but rather to illustrate that the sufficiency arguments that we presented translate into quantifiable performance differences in the deep RL setting.

2.5.1 *Experimental Setup*

To separate representation learning from RL, we first optimize each representation learning objective on a dataset of offline data consisting of 50k transitions collected from a uniform random policy. We then freeze the weights of the state encoder learned in the first phase and train RL agents with the representation as state input. To clearly illustrate the characteristics of each objective, we use the simple pygame (Shinners, 2011) video game *catcher*, in which the agent controls a paddle that it can move back and forth to catch fruit that falls from the top of the screen (see Figure 4). A positive reward is given when the fruit is caught and a negative reward when the fruit is not caught. The

episode terminates after one piece of fruit falls. We optimize $\mathbb{J}_{\text{fwd}}$ and $\mathbb{J}_{\text{state}}$ with noise contrastive estimation (Gutmann and Hyvärinen, 2010), and $\mathbb{J}_{\text{inv}}$ by training an inverse model via maximum likelihood. For the RL algorithm, we use the Soft Actor-Critic algorithm Haarnoja et al., 2018, modified slightly for the discrete action distribution. Please see Appendix A.2 for full experimental details.

2.5.2 *Computational Results*

In principle, we expect that a representation learned with $\mathbb{J}_{\text{inv}}$ may not sufficient to solve the *catcher* game. Because the agent does not control the fruit, a representation maximizing $\mathbb{J}_{\text{inv}}$ might discard that information, thereby making it impossible to represent the optimal policy. We observe in Figure 5 (top left) that indeed representations trained to maximize $\mathbb{J}_{\text{inv}}$ result in RL agents that converge slower and to a lower asymptotic expected return. Further, attempting to learn a decoder from the learned representation to the position of the falling fruit incurs a high error (Figure 5, bottom left), indicating that the fruit is not precisely captured by the representation. We argue that this type of problem setting is not contrived, and is representative of many situations in realistic tasks. Consider, for instance, an autonomous vehicle that is stopped at a stoplight. Because the agent does not control the color of the stoplight, it may not be captured in the representation learned by $\mathbb{J}_{\text{inv}}$ and the resulting RL policy may choose to run the light.

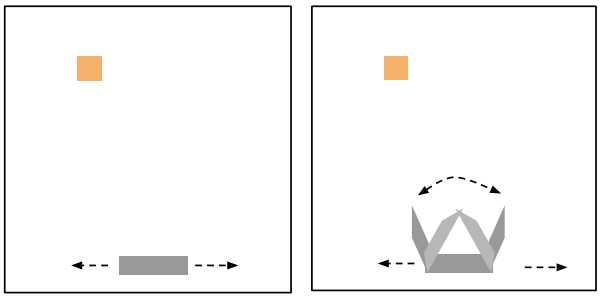
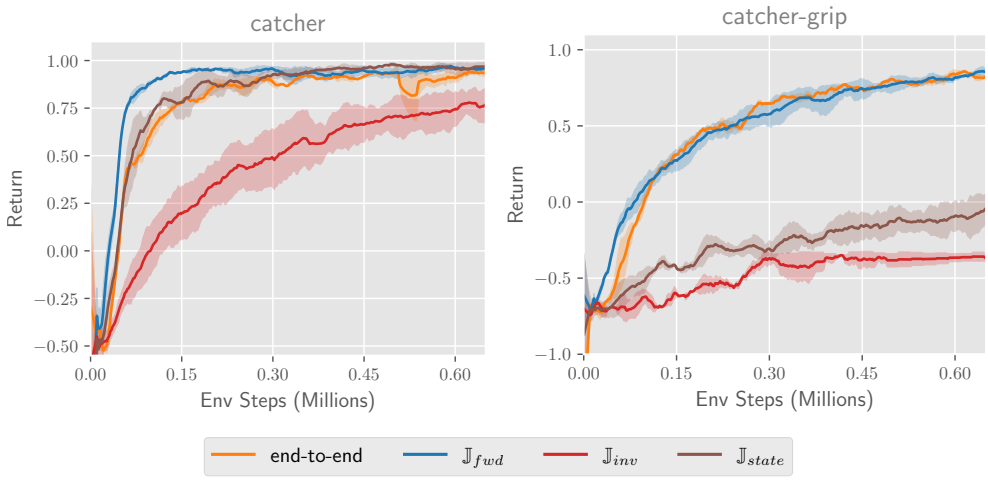


Figure 4: (left) Original *catcher* game in which the agent (grey paddle) moves left or right to catch fruit (yellow square) that falls from the top of the screen. (right) Variation *catcher-grip* in which the agent is instantiated as a gripper, and must open the gripper to catch fruit.

In the second experiment, we consider a failure mode of $\mathbb{J}_{\text{state}}$. We augment the paddle with a gripper that the agent controls and must be open in order to properly catch the fruit. Since the change in the gripper is completely controlled by a single action, the current state contains no information about the state of the gripper in the future. Therefore, a representation maximizing $\mathbb{J}_{\text{state}}$ might alias states where the gripper is open with states where the gripper is closed. In our experiment, we see that the error

in predicting the state of the gripper from the representation learned via $\mathbb{J}_{state}$ is chance (Figure 5, bottom right). This degrades the performance of an RL agent trained with this state representation since the best the agent can do is move under the fruit and randomly open or close the gripper (Figure 5, top right). In the driving example, suppose turning on the headlights incurs positive reward if it's raining but negative reward if it's sunny. The representation could fail to distinguish the state of the headlights, making it impossible to learn when to properly use the headlights.



Obj.	Agent Err.	Fruit Err.
$\mathbb{J}_{\text{fwd}}$	0.1	0.1
$\mathbb{J}_{\text{inv}}$	0.15	0.47

Obj.	Agent Err.	Fruit Err.	Grip Err.
$\mathbb{J}_{\text{fwd}}$	0.1	0.1	0.3
$\mathbb{J}_{\text{state}}$	0.1	0.1	0.5

Figure 5: (top) Policy performance using learned representations as state inputs to RL, for the *catcher* and *catcher-grip* environments. (bottom) Error in predicting the positions of ground truth state elements from each learned representation. Representations maximizing $\mathbb{J}_{\text{inv}}$ need not represent the fruit, while representations maximizing $\mathbb{J}_{\text{state}}$ need not represent the gripper, leading these representations to perform poorly in *catcher* and *catcher-grip* respectively.

$\mathbb{J}_{\text{fwd}}$ produces useful representations in all cases, and is equally or more effective than learning representations purely from the RL objective alone (as in Figure 5). We experiment with more visual complexity by adding background distractors; these results are presented in Appendix A.2.2. We find that in this setting representations learned with $\mathbb{J}_{\text{fwd}}$ to yield even larger gains over learning representations end-to-end via RL. We also

analyze the learned representations by evaluating how well they predict the optimal Q^* in Appendix A.2.1.

2.6 DISCUSSION

In this chapter, we aimed to analyze mutual information representation learning objectives for control from a theoretical perspective. In contrast to much prior work that studies how these objectives can be effectively optimized given high-dimensional observations, we analyze which objectives are guaranteed to yield representations that are actually sufficient for learning the optimal policy. Surprisingly, we show that two common objectives yield representations that are theoretically insufficient, and provide a proof of sufficiency for a third. We validate our theoretical results with an empirical investigation on a simple video game environment, and show that the insufficiency of these objectives can degrade the performance of deep RL agents.

We view this investigation as a step forward in understanding the theoretical characteristics of representation learning techniques commonly used in deep RL. We see many exciting avenues for future work. First, identifying more restrictive MDP classes in which insufficient objectives are in fact sufficient, and relating these to realistic applications. Second, investigating if sample complexity bounds can be established in the case of a sufficient objective. Third, extending our analysis to the partially observed setting, which is more reflective of practical applications. We see these directions as fruitful in providing a deeper understanding of the learning dynamics of deep RL, and potentially yielding novel algorithms for provably accelerating RL with representation learning.

3

META-LEARNING PROBLEM STATEMENT

In this chapter we define and formalize the meta-learning problem. Broadly defined, a meta-learning agent is an agent whose performance at a task improves not only with experience, but also with the number of *tasks* being learned (Thrun and Pratt, 1998). A task $\mathcal{T}$ can be any decision-making problem, from predicting the category of an object in an image to controlling an autonomous vehicle. In the next section, we formalize meta-learning for supervised learning problems, while in Section 3.2 we apply meta-learning to reinforcement learning.

3.1 SUPERVISED META-LEARNING

Let X represent the space of data points, Y the space of labels, $\mathcal{D} = \{(x, y)_n\}$ the available data, and ϕ the parameters of the model. A task is distinguished by a data distribution $p(x)$, an (unknown) distribution mapping data points to labels $p(y|x)$, and a loss function $\mathcal{L}(\phi, \mathcal{D})$. Formally then, a task $\mathcal{T}$ is defined as the tuple $\tau = (\mathcal{L}(\theta, \mathcal{D}), p(x), p(y|x))$. We assume that tasks are drawn from a distribution $p(\mathcal{T})$, and that all tasks in the distribution share input spaces but may have different output spaces. For example, the task distribution may be all possible semantic segmentations of objects in natural images. The input space is the space of natural images, and the output space is the space of possible pixel-wise labels for any number of objects or concepts. Note that this differs from conventional few-shot learning settings described in prior work (Vinyals et al., 2016; Finn et al., 2017a) which assume a fixed "shot" (how many training examples per class) and "way" (the number of classes to be distinguished). We argue that all few-shot learners should be able to handle varying shot and way, since in realistic applications it is rare that tasks would all have the same shot and way; for example, one might want to add a new class to an existing classifier.

Modern meta-learning methods work by casting the meta-learning problem into a

supervised learning problem, in which tasks (each one itself a supervised learning problem) take the place of data points. To see this, consider a standard discriminative machine learning method that trains a model $q_\phi(y|x)$ on training data $\mathcal{D}^{train} = \{(x^{tr}, y^{tr})_n\}$. In meta-learning, we have access to a set of meta-training tasks sampled from the task distribution $p(\tau)$. For each task i, we split the data available into $\mathcal{D}_i^{train}$ and $\mathcal{D}_i^{test}$. A discriminative meta-learning method would then learn the function, $q_\phi(y|\mathcal{D}_i^{train}, x)$, where $(x, y) \sim \mathcal{D}_i^{test}$. The resulting function q_ϕ can thus adapt the model to a *new* task given a small amount of labeled data.

While this framing illuminates the connection to standard supervised learning, it obscures how $\mathcal{D}_i^{train}$ is used for adaptation. Instead, we will write the meta-learning optimization problem as consisting of two loops: an inner adaptation loop that summarizes $\mathcal{D}_i^{train}$ into an adaptation procedure z_i that adapts the model, and an outer meta-training loop that maximizes the performance of the adapted model with respect to the parameters ϕ of the inner loop across the set of meta-training tasks.

$$\phi^* = \max_\phi \sum_{i=0}^{N} \mathcal{L}(z_i, \mathcal{D}_i^{test}) \text{ where } z_i = f_\phi(\mathcal{D}_i^{train}) \tag{5}$$

While the meta-training objective is typically optimized with gradient descent, many options are possible for the design of the adaptation procedure – the only requirement for our purposes is that it must be differentiable. The many designs that have been proposed for a myriad of applications can be broadly classified into two categories of approaches. *Context-based* methods summarize $\mathcal{D}_i^{train}$ into a latent vector z_i, and then condition the predictive model on z_i, often by feeding it as an input. Within the category of context-based methods, we can distinguish between black-box approaches that employ a recurrent network (Ravi and Larochelle, 2017) or other memory mechanism (Santoro et al., 2016), and those that make additional assumptions such as learning a metric space (Vinyals et al., 2016; Snell et al., 2017; Sung et al., 2018). *Gradient-based* methods (Finn et al., 2017a) interpret f_ϕ as several steps of gradient descent on the model parameters that adapts them to the task (in this case z_i can be interpreted as the adapted model parameters for task i, with pre-adapted parameters ϕ).

Gradient-based methods enjoy the benefit of consistency, meaning that as the amount of data used for adaptation increases, the meta-learner will converge on the same solution as if the task had been learned from scratch. However, particularly for small amounts of data, context-based methods are often faster, easier to optimize, and achieve higher accuracy (Rakelly et al., 2019; Ren et al., 2020). The best approach often depends on the application. For applications such as structured output prediction considered in Chap-

ter 4, very large convolutional networks that have millions of parameters are required, making a gradient-based approach extremely slow. Additionally, due to memory constraints, these models are typically trained with very small batch sizes, which when combined with gradient-based meta-learning can result in thrashing between parameter settings rather than converging to a solution. In Chapter 4, we propose a context-based meta-learning method for meta-learning few-shot image segmentation.

3.2 META-REINFORCEMENT LEARNING (META-RL)

In this section we specialize the meta-learning setup to the case of reinforcement learning and point out differences from the supervised learning problems we have studied so far. In meta-reinforcement learning (meta-RL), each task is a Markov Decision Process (MDP). An MDP consists of a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, an initial state distribution $p(\mathbf{s}_1)$, a state transition distribution $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, a discount factor γ, and a reward function $r(\mathbf{s}_t, \mathbf{a}_t)$. We assume the transition and reward functions are unknown, but can be sampled by taking actions in the environment. The goal of RL is to learn a policy $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ that selects actions that maximize the sum of discounted rewards.

While a review of RL algorithms is beyond the scope of this work, one characteristic of RL algorithms that will be important in later chapters is the type of data required for learning. RL algorithms can be categorized into those that learn from data collected by the current policy ("on-policy"), and those that can learn from data collected by a different policy ("off-policy") (Sutton et al., 1998). Off-policy algorithms are substantially more data efficient, as they can re-use previously collected data for training. Since deep RL is data-intensive, often requiring millions of billions of environment interactions to learn a good policy, we will be interested in leveraging off-policy learning wherever possible to reduce the burden of interaction.

In the meta-RL setting, we assume a distribution over tasks $p(\mathcal{T})$. This problem definition encompasses task distributions with varying transition functions (e.g., robots with different dynamics) and varying reward functions (e.g., navigating to different locations). We formalize the adaptation procedure f_ϕ as a function of experience $(\mathbf{s}_{1:t}, r_{1:t}, \mathbf{a}_{1:t-1})$ that summarizes task-relevant information into the variable $\mathbf{z}_t$. The policy is conditioned on this variable as $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t, \mathbf{z}_t)$ to adapt to the task. By training the adaptation mechanism f_ϕ and the policy π_θ end-to-end to maximize returns of the adapted policy, meta-RL algorithms can learn policies that effectively modulate and adapt their behavior with

small amounts of experience in new tasks. We formalize this meta-RL objective as:

$$\theta^*, \phi^* = \max_{\theta,\phi} \mathop{\mathbb{E}}_{\mathcal{T}\sim p(\mathcal{T})} \mathop{\mathbb{E}}_{\substack{a_t\sim\pi_\theta(\cdot|s_t,z_t)\\ s_{t+1}\sim p_{\mathcal{T}}(\cdot|s_t,a_t)\\ r_t\sim r_{\mathcal{T}}(\cdot|s_t,a_t)}} \left[\sum_{t=1}^{T} \gamma^t r_t\right] \quad \text{where} \quad z_t = f_\phi(s_{1:t}, r_{1:t}, a_{1:t-1}). \tag{6}$$

Similar to the supervised learning setting, prior meta-RL methods differ in how the adaptation procedure f_ϕ is represented: as a recurrent update (Duan et al., 2016; Wang and Hebert, 2016), or as a gradient step (Finn et al., 2017a)). Additionally, meta-RL algorithms also differ in how often the adaptation procedure occurs (e.g., at every timestep (Duan et al., 2016; Zintgraf et al., 2019) or once per episode (as in the PEARL algorithm proposed in Chapter 5 as well as Humplik et al. (2019)), and in how the optimization is performed (e.g., on-policy (Duan et al., 2016), off-policy as in both algorithms proposed in Chapter 5). A significant difference between the meta-RL and supervised meta-learning settings is that in meta-RL the agent must collect its own adaptation data. Thus adaptation to a new task presents the same exploration-exploitation problem inherent to RL. During the meta-training phase, the agent must learn both exploration strategies and how to make use of the collected data to learn the task.

In Chapter 5, we tackle combining meta-RL with off-policy learning to produce the sample efficient meta-RL algorithm PEARL. To do so, we must contend with the exploration problem, since in the off-policy setting, the data seen during meta-training is systematically different from the data collected when adapting to a new task. As we will see, framing meta-RL as probabilistic inference of task variables affords an elegant solution to this exploration problem.

3.3 PARTIALLY OBSERVED META-RL

Robots operating in the real world do not have access to the underlying state s_t, and must instead select actions using high-dimensional and often incomplete observations from cameras and other sensors. Such a system can be described as a partially observed Markov decision process (POMDP), where observations x_t are a noisy or incomplete function of the unknown underlying state s_t, and the policy is conditioned on a history of observations as $\pi(a_t|x_{1:t})$. In the meta-RL setting, each $\mathcal{T}$ from a distribution of tasks $p(\mathcal{T})$ is now a POMDP as described above, with initial state distribution $p_{\mathcal{T}}(s_1)$, dynamics function $p_{\mathcal{T}}(s_{t+1}|s_t, a_t)$, observation function $p_{\mathcal{T}}(x_t|s_t)$, and reward function $r_{\mathcal{T}}(r_t|s_t, a_t)$. The partially observed meta-RL objective is very similar to Equation 12 (additions are

marked in blue):

$$\theta^*, \phi^* = \max_{\theta,\phi} \mathop{\mathbb{E}}_{\mathcal{T}\sim p(\mathcal{T})} \mathop{\mathbb{E}}_{\substack{\mathbf{x}_t\sim p_{\mathcal{T}}(\cdot|\mathbf{s}_t) \\ \mathbf{a}_t\sim\pi_\theta(\cdot|\mathbf{x}_{1:t},\mathbf{z}_t) \\ \mathbf{s}_{t+1}\sim p_{\mathcal{T}}(\cdot|\mathbf{s}_t,\mathbf{a}_t) \\ r_t\sim r_{\mathcal{T}}(\cdot|\mathbf{s}_t,\mathbf{a}_t)}} \left[\sum_{t=1}^{T} \gamma^t r_t\right] \quad \text{where} \quad \mathbf{z}_t = f_\phi(\mathbf{x}_{1:t}, r_{1:t}, \mathbf{a}_{1:t-1}). \tag{7}$$

Black-box recurrent meta-RL algorithms can handle such state partial observability by design (the function f and the policy π are both the same recurrent network). However these approaches have a number of drawbacks: they provide no explicit representation learning to handle image inputs, they do not represent uncertainty in either task or state estimation, and it has proven difficult to use such models in conjunction with off-policy learning (see Chapter 5).

In Chapter 5, we again leverage the probabilistic inference perspective on meta-learning to cast meta-RL into the framework of latent state estimation. This insight allows us to leverage latent state models designed to infer state information from history directly for meta-RL with only a few modifications. The resulting MELD algorithm uses unsupervised representation learning to aid in interpreting image inputs, represents uncertainty over both the state and task, and is amenable to off-policy optimization.

4

GUIDING IMAGE SEGMENTATION VIA META-LEARNING

Many tasks of scientific and practical interest require grouping pixels, such as cellular microscopy, medical imaging, and graphic design. Furthermore, a single image might need to be segmented in several ways, for instance to first segment all people, then focus on a single person, and finally pick out their face. Learning a particular type of segmentation, or even extending an existing model to a new task like a new semantic class, generally requires collecting and annotating a large amount of data and (re-)training a large model for many iterations. Interactive segmentation with a supervisor in-the-loop can cope with less supervision such as a few clicks on the image, but the concepts indicated by the annotations cannot be transferred to other images. Faced with endless varieties of segmentation and countless images, and limited annotator expertise and time, a segmentor should be able to learn from varying amounts of supervision and propagate that supervision to unlabeled pixels and images. We refer to this problem statement as *guided* segmentation. The amount of supervision may vary widely, from a lone annotated pixel, millions of pixels in a fully annotated image, or even more across a collection of images as in conventional supervised learning for segmentation. The number of classes to be segmented may also vary depending on the task, such as when segmenting categories like cats vs. dogs, or when segmenting instances to group individual people. Guided segmentation extends few-shot learning to the structured output setting, and the non-episodic accumulation of supervision as data is progressively annotated. Compared to interactive segmentation, guided segmentation broadens the scope by integrating supervision across images and segmenting unannotated images.

We frame the problem of guided segmentation as a meta-learning problem, where tasks consist of different concepts to segment. To tackle this problem, we propose an algorithm that learns to infer a latent task variable from the supervision available, and conditions the segmentation network on this latent variable in order to segment new images accordingly (see Figure 6). We perform meta-training by synthesizing few-shot

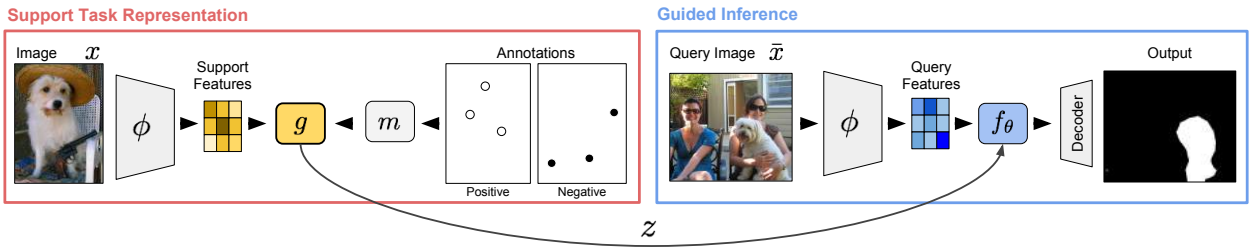


Figure 6: A guide g extracts a latent task representation z from an annotated image (red) for inference by $f_\theta(\bar{x}, z)$ on a different, unannotated image (blue).

tasks from fully labeled segmentation datasets. Once trained, our model can quickly and cumulatively incorporate annotations to perform new tasks not seen during training. Guided networks reconcile static and interactive modes of inference: a guided model is both able to make predictions on its own, like a fully supervised model, and to incorporate expert supervision for defining new tasks or correcting errors, like an interactive model.

We evaluate our method on a variety of challenging segmentation problems in Section 4.4: interactive image segmentation, semantic segmentation, video object segmentation, and real-time interactive video segmentation, as shown in 7. We compare guided networks with standard supervised learning across the few-shot and many-shot extremes to identify the boundary between few-shot and many-shot learning for segmentation. Finally, we demonstrate that in some cases, our model can generalize to guide tasks at a different level of granularity, such as meta-learning from instance supervision and then guiding semantic segmentation of categories.

4.1 RELATED WORK

Guided segmentation extends few-shot learning to structured output models, statistically dependent data, and variable supervision in amount of annotation (shot) and numbers of classes (way). Guided segmentation spans different kinds of segmentation as special cases determined by the supervision that constitutes a task, such as a collection of category masks for semantic segmentation, sparse positive and negative pixels in an image for interactive segmentation, or a partial annotation of an object on the first frame of a clip for video object segmentation.

Few-shot learning Few-shot learning (Fei-Fei et al., 2006; B. M. Lake et al., 2015) holds the promise of data efficiency: in the extreme case, one-shot learning requires

only a single annotation of a new concept. The present wave of methods (Koch et al., 2015; Santoro et al., 2016; Vinyals et al., 2016; Wang and Hebert, 2016; Bertinetto et al., 2016; Finn et al., 2017a; Ravi and Larochelle, 2017; Snell et al., 2017) frame it as direct optimization for the few-shot setting: they synthesize episodes by sampling supports and queries, define a task loss, and learn a task model for inference of the queries given the support supervision. While these works address a setting with a fixed, small number of examples and classes at meta-test time, we explore settings where the number of annotations and classes is flexible.

Our approach is most closely related to episodically optimized metric learning approaches. We design a novel, efficient segmentation architecture for metric learning, inspired by Siamese networks (Chopra et al., 2005; Hadsell et al., 2006) and few-shot metric methods (Koch et al., 2015; Vinyals et al., 2016; Snell et al., 2017) that learn a distance to retrieve support annotations for the query. In contrast to existing meta-learning schemes, we examine how a meta-learned model generalizes across task families with a nested structure, such as performing semantic segmentation after meta-learning on instance segmentation tasks.

Segmentation There are many kinds of segmentation, and many current directions for deep learning techniques (Garcia-Garcia et al., 2017). We take up semantic (Everingham et al., 2010; Liu et al., 2011), interactive (Kass et al., 1988; Boykov and Jolly, 2001), and semi-supervised video object segmentation (Pont-Tuset et al., 2017) as challenge problems for our unified view with guidance. See Fig. 7 for summaries of these tasks.

For semantic segmentation Shaban et al. (2017) proposes a one-shot segmentor (OSLSM), which requires few but densely annotated images, and must independently infer one annotation and class at a time. Our guided segmentor can segment from sparsely annotated pixels and perform multi-way inference. For video object segmentation one-shot video object segmentation (OSVOS) by Caelles et al. (2017) achieve high accuracy by fine-tuning during inference, but this online optimization is too costly in time and fails with sparse annotations. Our guided segmentor is feed-forward, hence quick, and segments more accurately from extremely sparse annotations. Y. Chen et al. (2018) impressively achieve state-of-the-art accuracy and real-time, interactive video object segmentation by replacing online optimization with offline metric learning and nearest neighbor inference on a deep, spatiotemporal embedding; however, they focus exclusively on video segmentation. We consider a variety of segmentation tasks, and investigate how guidance transfers across semantic and instance tasks and how it scales with more annotation. For interactive segmentation, N. Xu et al. (2016); Maninis et al. (2018) learn state-of-the-art interactive object segmentation, and Maninis et al. (2018) only needs four annotations per

Figure 7: Guided segmentation groups different kinds of segmentation in one problem statement.

object. However, these purely interactive methods infer each task in isolation and cannot pool supervision across tasks and images without optimization, while our guided segmentor quickly propagates supervision non-locally between images.

4.2 GUIDED SEGMENTATION

Akin to few-shot learning, we divide the input into an annotated support, which supervises the task to be done, and an unannotated query on which to do the task. The common setting in which the support contains K distinct classes and S examples of each is referred to as K-way, S-shot learning (B. M. Lake et al., 2015; Fei-Fei et al., 2006; Vinyals et al., 2016). For guided segmentation tasks we add a further pixel dimension to this setting, as we must now consider the number of support pixel annotations for each image, as well as the number of annotated support images. We denote the number of annotated pixels per image as P, and consider the settings of (S, P)-shot learning for various S and P. In particular, we focus on sparse annotations where P is small, as these are more practical to collect, and merely require the annotator to point to the segment(s) of interest. This type of data collection is more efficient than collecting dense masks by at least an order of magnitude (Bearman et al., 2016). Since segmentation commonly has imbalanced classes and sparse annotations, we consider mixed-shot and semi-supervised supports where the shot varies by class and some points are unlabeled. This is in contrast to the standard few-shot assumption of class-balanced supports.

We define a guided segmentation task as the set of input-output pairs $(\mathcal{T}_i, \mathcal{Y}_i)$ sampled from a task distribution $\mathcal{P}$, adopting and extending the notation of Garcia and Bruna (2018). The task inputs are

$$\mathcal{T} = \left\{ \{(x_1, L_1), \dots (x_S, L_S)\} \cup \{\bar{x}_1, \dots, \bar{x}_Q\} \; ; \; x_s, \bar{x}_q \sim \mathcal{P}_l(\mathbb{R}^N) \right\}$$
$$L_s = \left\{ (p_j, l_j) : j \in \{1 \dots P\}, \; l \in \{1 \dots K\} \cup \{\varnothing\} \right\}$$

where S is the number of annotated support images x_s, Q is the number of unannotated query images $\bar{x}_q$, and L_s are the support annotations. The annotations are sets of point-label pairs (p, l) with $|L_s| = P$ per image, where every label l is one of the K classes or unknown ($\varnothing$). The task outputs, that is the targets for the support-defined segmentation task on the queries, are

$$y = (y_1, \ldots, y_Q), \quad y_q = \left\{(p_j, l_j) : p_j \in \bar{x}_q\right\}$$

Our model handles general way K, but for exposition we focus on binary tasks with $K = 2$, or $L = (+, -)$. We let $Q = 1$ throughout as inference of each query is independent in our model.

4.3 GUIDED NETWORKS

Our approach to guided segmentation has two parts: (1) extracting a task representation from the semi-supervised, structured support and (2) segmenting the query given the task representation. We define the task representation as $z = g(x, +, -)$, and the query segmentation guided by that representation as $\hat{y} = f(\bar{x}, z)$. The design of the task representation z and its encoder g is crucial for guided segmentation to handle the hierarchical structure of images and pixels, the high and variable dimensions of images and their pixelwise annotations, the semi-supervised nature of support with many unannotated pixels, and skewed support distributions.

We examine how to best design the guide g and inference f as deep networks. Our method is one part architecture and one part optimization. For architecture, we define branched fully convolutional networks, with a guide branch for extracting the task representation from the support with a novel late fusion technique (Section 4.3.1), and an inference branch for segmenting queries given the guidance (Section 4.3.2). For optimization, we adapt episodic meta-learning to image-to-image learning for structured output (Section 4.3.3), and increase the diversity of episodes past existing practice by sampling within and *across* segmentation task families like categories and instances.

4.3.1 *Guidance: From Support to Latent Task Representation*

The task representation z must fuse the visual information from the image with the annotations in order to determine what should be segmented in the query. As images with (partial) segmentations, our support is statistically dependent because pixels are spatially correlated, semi-supervised because the full supervision is arduous to annotate,

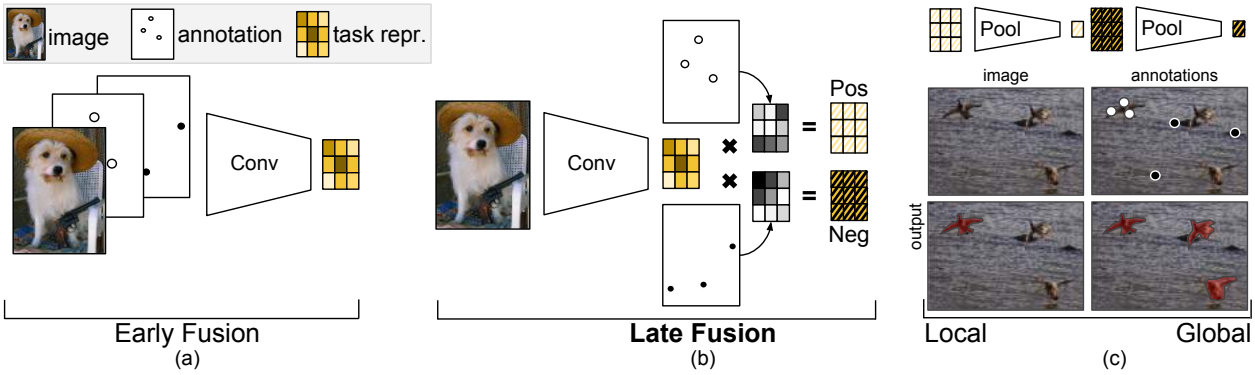


Figure 8: Extracting a task representation or "guidance" from the support. (a) Early fusion simply concatenates the image and annotations. (b) Our late fusion factorizes into image and annotation streams, improves accuracy, and updates quickly given new annotations. (c) Globalizing the task representation propagates appearance non-locally: a single bird is annotated in this example, but global guidance causes all the similar-looking birds to be segmented (red) regardless of location.

and high dimensional and class-skewed because scenes are sizable and complicated. For simplicity, we first consider a binary task with $(1, P)$-shot support consisting of one image with an arbitrary number of annotated pixels P, and then extend to multi-way tasks and general (S, P)-shot support. To begin we decompose the support encoder $g(x_s, +_s, -_s)$ across receptive fields indexed by i for local task representations $z_i = g(x_{si}, +_{si}, -_{si})$; this is the same independence assumption made by existing fully convolutional approaches to structured output. See Figure 8 for an overview and our novel late global fusion technique.

Early Fusion (prior work) Stacking the image and annotations channel-wise at the input makes $z_{si} = g_{\text{early}}(x, +, -) = \phi_S(x \oplus + \oplus -)$ with a support feature extractor ϕ_S. This early fusion strategy, employed by N. Xu et al., 2016, gives end-to-end learning full control of how to fuse. Masking the image by the positive pixels (Shaban et al., 2017; J. S. Yoon et al., 2017) instead forces invariance to context, potentially speeding up learning, but precludes learning from the background and disturbs input statistics. All early fusion techniques suffer from an inherent modeling issue: incompatibility of the support and query representations. Stacking requires distinct ϕ_S, ϕ_Q while masking disturbs the input distribution. Early fusion is slow, since changes in annotations trigger a full pass through the network, and only one task can be inferred at a time, limiting existing interactive and few-shot segmentors alike (N. Xu et al., 2016; Maninis et al., 2018; Shaban et al., 2017).

Late Fusion (ours) We resolve the learning and inference issues of early fusion by factorizing features and annotations in the guide architecture as $z_{si} = g_{late}(x, +, -) = \psi(\phi(\bar{x}), m(+), m(-))$. We first extract visual features from the image alone by $\phi(x)$, map the annotations into masks in the feature layer coordinates $m(+), m(-)$, and then fuse both by ψ chosen to be element-wise product. This factorization into visual and annotation branches defines the spatial relationship between image and annotations, improving learning sample efficiency and inference computation time. Fixing m to interpolation and ψ to multiplication, the task representation can be updated quickly by only recomputing the masking and not features ϕ. See Figure 8 (center). We do not model a distribution over z, although this is a possible extension of our work for regularization or sampling diverse segmentations.

Our late fusion architecture can now share the feature extractor ϕ for joint optimization through the support and query. Sharing improves learning efficiency with convergence in fewer iterations and task accuracy with 60% relative improvement for video object segmentation. Late fusion reduces inference time, as only the masking needs to be recomputed to incorporate new annotations, making it capable of real-time interactive video segmentation. Optimization-based methods (Caelles et al., 2017) need seconds or minutes to update.

Locality We are generally interested in segmentation tasks that are determined by visual characteristics and not absolute location in space or time, i.e. the task is to group pixels of an object and not pixels in the bottom-left of an image. When the support and query images differ, there is no known spatial correspondence, and the only mapping between support and query should be through features. To fit the architecture to this task structure, we merge the local task representations by $m_P(\{z_{si} : \forall i\})$ for all positions i. Choosing global pooling for m_P globalizes the task by discarding the spatial dimensions. The pooling step can be done by averaging, our choice, or other reductions. The effect of pooling in an image with multiple visually similar objects is shown in Figure 8 (right).

Multi-Shot and Multi-Way The full (S, P)-shot setting requires summarizing the entire support with a variable number of images with varying amounts of pixelwise annotations. Note in this case that the annotations might be divided across the support, for instance one frame of a video may only have positives while a different frame has only negatives, so S-shot cannot always be reduced to 1-shot, as done in prior work Shaban et al., 2017. We form the full task representation $z_S = m_S(\{z_1, \ldots, z_S\})$ simply and differentiably by averaging the shot-wise representations z_s. While we have considered binary tasks thus far, we extend guidance to multi-way inference do in our experiments. We construct a separate guide for each class, averaging across all shots containing annota-

Figure 9: Optimization for guided segmentation. (a) Synthesizing tasks from densely annotated segmentation data. (b) One task update: episodic training reduces to supervised learning.

tions for that class. Note that all the guides share ϕ for efficiency and differ only in the masking.

4.3.2 *Guiding Inference*

Inference in a static segmentation model is simply $\hat{y} = f_\theta(\bar{x})$ for output y, parameters θ, and input $\bar{x}$. Guided inference is a function $\hat{y} = f(\bar{x}, z)$ of the query given the guidance extracted from the support. We further structure inference by $f(\phi(\bar{x}), z)$, where ϕ is a fully convolutional encoder from input pixels to visual features.

Multiple forms of conditioning are possible and have been explored for low-dimensional classification and regression problems by the few-shot learning literature. In preliminary experiments we consider parameter regression, nearest neighbor and prototype retrieval, and metric learning on fused features. We select metric learning with feature fusion because it was simple and robust to optimize. Note that feature fusion is similar to siamese architectures, but we directly optimize the classification loss rather than a contrastive loss.

In particular we fuse features by $m_f = \phi(x) \oplus \text{tile}(z)$ which concatenates the guide with the query features, while tiling z to the spatial dimensions of the query. The fused query-support feature is then scored by a small convolutional network f_θ that can be interpreted as a learned distance metric for retrieval from support to query. For multi-way guidance, the fusions of the query and each guide are batched for parallel inference.

4.3.3 *Episodic Optimization and Task Distributions*

We distinguish between optimizing the parameters of the model during training (learning) and adapting the model during inference (guidance). Thus during training, we wish to "learn to guide." In standard supervised learning, the model parameters θ are optimized according to the loss between prediction $\hat{y} = f_\theta(x)$ and target y. We reduce the problem of learning to guide to supervised learning by jointly optimizing the parameters of the guidance branch g and the segmentation branch f according to the loss between $f_\theta(\bar{x}, z)$ and query target y, see Figure 9.

For clarity, we distinguish between tasks, a given support and query for segmentation, and task distributions that define a kind of segmentation problem. For example, semantic segmentation is a task distribution while segmenting birds (a semantic class) is a task. We train a guided network for each task distribution by optimizing episodically on sampled tasks. The supports and queries that comprise an episode are synthesized from a fully labeled dataset. We first sample a task, then a subset of images containing that task which we divide into support and query. During training, the target for the query image is available, while for testing it is not. We binarize support and query annotations to encode the task, and spatially sample support annotations for sparsity.

Given inputs and targets, we train the network with the pixelwise cross-entropy loss between the predicted and target segmentation of the query. See Sections B.1 and B.1.1 for more details on data processing and network optimization respectively.

After learning, the model parameters are fixed, and task inference is determined by guidance. While we evaluate for varying support size S, as described in 4.3.2, we train with $S = 1$ for efficiency while sampling $P \sim \text{Uniform}(1, 100)$. Once learned, our guided networks can operate at different (S, P) shots to address sparse and dense pixelwise annotations with the same model, unlike existing methods that train for particular shot and way. In our experiments, we train with tasks sampled from a single task distribution, but co- or cross-supervision of distributions is possible. Intriguingly, we see some transfer between distributions when evaluating a guided network on a different distribution than it was trained on in Section 4.4.3.

4.4 RESULTS

We evaluate our guided segmentor on a variety of problems that are representative of segmentation as a whole: interactive segmentation, semantic segmentation, and video object segmentation. These are conventionally regarded as separate problems, but we

demonstrate that each can be viewed as an instantiation of guided segmentation. As a further demonstration of our method, we present results for real-time, interactive video segmentation from dot annotations. To better understand the characteristics of guidance, we experiment with cross-task supervision in Section 4.4.2 and guiding with large-scale supports in Section 4.4.3.

To standardize evaluation we select one metric for all tasks: the intersection-over-union (IU) of the positives averaged across all tasks and masks. This choice allows us to compare scores across the different kinds of segmentation we consider without skew from varying numbers of classes or images. Note that this metric is not equivalent to the mean IU across classes that is commonly reported for semantic segmentation. Please refer to Section B.1.2 for more detail.

We include fine-tuning and foreground-background segmentation as baselines for all problems. Fine-tuning simply attempts to optimize the model on the support. Foreground-background verifies that methods are learning to co-vary their output with the support supervision and sets an accuracy floor.

The backbone of our networks is VGG-16 (Simonyan and Zisserman, 2015), pre-trained on ILSVRC Russakovsky et al., 2015, and cast into fully convolutional form (Shelhamer et al., 2016a). This choice is made for fair comparison with existing works across our challenge tasks of semantic, interactive, and video object segmentation without confounds of architecture, pre-training data, and so forth.

4.4.1 *Guidance for Interactive, Video Object, and Semantic Segmentation*

Interactive Image Segmentation We recover this problem as a special case of guided segmentation when the support and query images are identical. We evaluate on PASCAL VOC (Everingham et al., 2010) and compare to deep interactive object selection (DIOS) (N. Xu et al., 2016), because it is state-of-the-art and shares our focus on learning for label efficiency and generality. Our approach differs in support encoding: DIOS fuses early while we fuse late and globally. Our guided segmentor is more accurate with extreme sparsity and intrinsically faster to update, as DIOS must do a full forward pass. See Figure 10 (left). From this result we decide on late-global guidance throughout.

Video Object Segmentation We evaluate our guided segmentor on the DAVIS 2017 benchmark (Pont-Tuset et al., 2017) of 2–3 second videos. For this problem, the object indicated by the fully annotated first frame must be segmented across the video. We then extend the benchmark to sparse annotations to gauge how methods degrade. We compare to OSVOS (Caelles et al., 2017), a state-of-the-art online optimization method

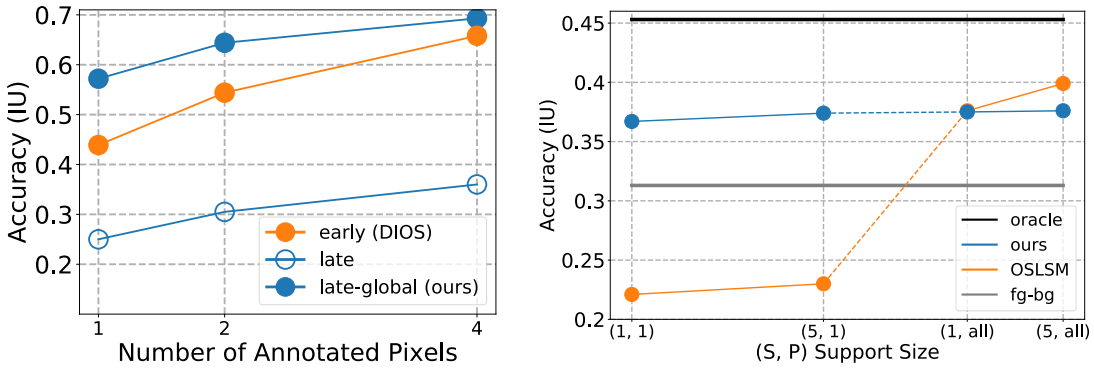


Figure 10: (left) Interactive segmentation of objects in images. (right) Guided semantic segmentation of held-out classes: we are state-of-the-art with only two points and competitive with full annotations.

that fine-tunes on the annotated frame and then segments the video frame-by-frame. While Y. Chen et al. (2018) presents impressive results on this task and on real-time interactive video segmentation without optimization, their scope is limited to video, and they employ orthogonal improvements that make comparison difficult. We were unable to reproduce their results in our own experimental framework. See Figure 11 (left) for a comparison of accuracy, speed, and annotation sparsity.

In the dense regime our method achieves 33.3% accuracy for 80% relative improvement over OSVOS in the same time envelope. Given (much) more time fine-tuning significantly improves in accuracy, but takes 10+min/video. Guidance is $\sim 200\times$ faster at 3sec/video. Our method handles extreme sparsity with little degradation, maintaining 87% of the dense accuracy with only 5 points for positive and negative. Fine-tuning struggles to optimize over so few annotations.

Interactive Video Segmentation By dividing guidance and inference, our guided segmentor can interactively segment video in real time. As an initial evaluation, we simulate interactions with randomly-sampled dot annotations. We define a benchmark by fixing the amount of annotation and measuring accuracy as the annotations are given. The accuracy-annotation tradeoff curve is plotted in Figure 11 (right). Our guided segmentor improves with both dimensions of shot, whether images (S) or pixels (P). Our guided architecture is feedforward and fast, and faster still to update for changes to the annotations.

Semantic Segmentation Semantic segmentation is a challenge for learning from little data due to the high intra-class variance of appearance. For this problem it is crucial

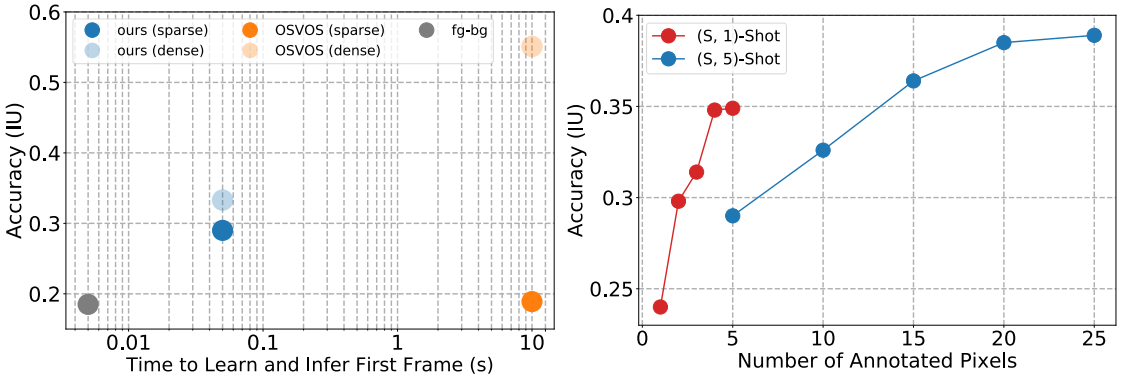


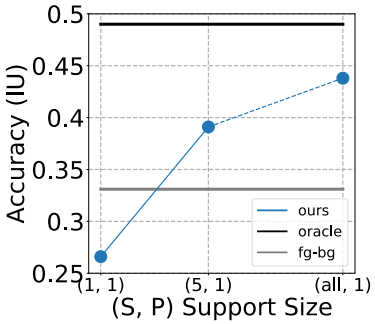
Figure 11: (left) Accuracy-time evaluation for sparse and dense video object segmentation on DAVIS'17 val. (right) Real-time interactive video segmentation on simulated dot interactions.

to evaluate on not only held-out inputs, but held-out classes, to be certain the guided learner has not covertly learned to be an unguided semantic segmentor. To do so we follow the experimental protocol of Shaban et al. (2017) and score by averaging across four class-wise splits of PASCAL VOC (Everingham et al., 2010), with has 21 classes (including background), and compare to OSLSM.

Our approach achieves state-of-the-art sparse results that rival the most accurate dense results with just two labeled pixels: see Figure 10 (right). OSLSM is incompatible with missing annotations, as it does early fusion by masking, and so is only defined for $\{0, 1\}$ annotations. To evaluate it we map all missing annotations to negative. Foreground-background is a strong baseline, and we were unable to improve on it with fine-tuning. The oracle is trained on all classes (nothing is held-out).

4.4.2 *Guiding Classes from Instances*

We carry out a novel examination of meta-learning with cross-task supervision. In the language of task distributions, the distribution of instance tasks for a given semantic category are nested in the distribution of tasks for that category. We investigate whether meta-training on the sub-tasks (instances) can address the super-tasks (classes). This tests whether guidance can capture an enumerative definition of a semantic class as the union of instances in that category.



To do so, we meta-train our guided segmentor on interactive *instance* segmentation tasks draw from all classes of PASCAL VOC (Everingham et al., 2010), and then evaluate the model on *semantic* segmentation tasks from all categories. We experiment with $(S, 1)$ support from semantic annotations, where S varies from one image to all the images in the training set, shown in the plot to the right. We compare to foreground-background as a class-agnostic accuracy floor, and a standard semantic segmentation net trained with semantic labels as an oracle. Increasing the amount of semantic annotations for guidance steadily increases accuracy.

4.4.3 *Guiding by Few or Many Annotations*

Thus far we have considered guidance in a variable but constrained scale of annotations, ranging from a single pixel in a single image to a few fully annotated images. We meta-learned our guided networks over episodes with such support sizes, and they perform accordingly well in this regime. Here we consider a much wider spectrum of support sizes, with the goal of understanding how guidance compares to standard supervised learning at both ends of the spectrum. To the best of our knowledge, this is the first evaluation of how few-shot learning scales to many-shot usage for structured output.

For this experiment we compare guidance and supervised learning on a transfer task between disjoint semantic categories. We take the classes of PASCAL VOC (Everingham et al., 2010) as source classes, and take the non-intersecting classes of COCO (Lin et al., 2014) as the target classes. We divide COCO 2017 validation into class-balanced train/test halves to look at transfer from a practical amount of annotation (thousands instead of more than a hundred thousand images). Our guided segmentor is meta-trained with semantic tasks sampled from the source classes, then guided with 5,989 densely annotated semantic masks from the target classes. For fair comparison, the supervised learner is first trained on the source classes, and then fine-tuned on the same annotated target data. Both methods share the same ILSVRC pre-training, backbone architecture, and (approximate) number of parameters. In this many-shot regime, guidance achieves 95% of supervised learning performance. A key point of this result is to shed light on the spectrum of supervision that spans few-shot and many-shot settings, and encourage future work to explore bridging the two.

4.5 DISCUSSION

From a computer vision perspective, the problem statement of guided segmentation unites interactive, few-shot, and video object segmentation in a unified framework for propagating annotations across image collections. Guided networks reconcile task-driven and interactive inference by extracting guidance, a latent task representation. The algorithm can learn to segment from a wide range of annotated pixels, from 1 to the millions of pixels in a fully annotated image, and can segment a variable number of concepts. The late fusion architecture that we propose to infer the latent task vector enables extremely fast inference; for example, in a video segmentation scenario, the image features can be pre-computed, allowing the segmentations to be updated in real time as the user adds annotations.

From the perspective of meta-learning, our proposed guided segmentor leverages a task inference approach to extend few-shot learning to the realm of structured output models. Unlike many few-shot classification algorithms (Vinyals et al., 2016; Finn et al., 2017a), our segmentor is agnostic to "shot" and "way." Still, if a new concept is too different from the concepts seen during meta-training, the guided segmentor can fail to learn to segment the new concept. Additionally, the performance of the algorithm saturates with a certain amount of supervision (25 labeled pixels in the video object segmentation experiment, Figure 11). In these cases, learning via gradient descent would enable the model to keep improving. While all model parameters could be trained, it would also be possible to train only the latent task vector. An algorithm that trades off smoothly between inference and gradient descent is an interesting direction of future work, for segmentation but also for meta-learning more broadly.

In this chapter we focused on agents that make passive predictions. Next, we turn to agents that interact with their environment and study the reinforcement learning setting. We will continue to build on the idea of meta-learning as task inference, extending it to the probabilistic setting.

5

EFFICIENT META-RL VIA PROBABILISTIC TASK INFERENCE

General purpose autonomous robots must be able to perform a wide variety of tasks and quickly acquire new skills. For example, consider a robot tasked with assembling electronics in a data center. This robot must be able to insert cables of varying shapes, sizes, colors, and weights into the correct ports with the appropriate amounts of force. While the combination of reinforcement learning (RL) with powerful non-linear function approximators has led to a wide range of advances in sequential decision making problems, conventional RL methods learn a separate policy per task, each often requiring millions of interactions with the environment. Learning large repertoires of behaviors with such methods quickly becomes prohibitive. Fortunately, many of the problems we would like our autonomous agents to solve share common structure; for example, all cable insertion tasks involve manipulating the cable with precision into the socket. As introduced in Chapter 3, meta-learning approaches offer an opportunity to exploit this structure to learn new tasks more quickly. Given a task distribution, such as the variety of ways to insert cables described above, meta-RL algorithms leverage a set of training tasks to meta-learn a mechanism that can quickly learn unseen tasks from the same distribution. Despite promising results in simulation demonstrating that agents can learn new tasks in a handful of trials (Wang and Hebert, 2016; Duan et al., 2016; Finn et al., 2017a; Rothfuss et al., 2018), during the meta-training phase these algorithms require massive amounts of data drawn from a large set of distinct tasks, exacerbating the problem of sample efficiency that plagues RL algorithms.

While meta-learning can be viewed as learning a learning algorithm (Wang and Hebert, 2016; Duan et al., 2016), or learning parameter values amenable to adaptation via gradient descent (Finn et al., 2017a; Rothfuss et al., 2018), an alternative perspective frames it as probabilistic inference of hidden task variables (Rusu et al., 2019; J. Gordon et al., 2019; Finn et al., 2018). Extending this view to the control setting, this perspective reveals that meta-RL is a special kind of partially observed MDP in which the hidden variable is the

task to be performed. Leveraging this insight, in this chapter we propose two meta-RL algorithms based on posterior belief inference given experience consisting of observations and rewards.

First, we tackle the problem of efficient off-policy meta-RL. In Section 5.2, we propose an algorithm PEARL that performs adaptation via variational posterior inference of hidden task variables and explores via posterior sampling. Disentangling task inference from action minimizes distribution mismatch between meta-train and meta-test, enabling off-policy meta-training; the policy can be optimized with off-policy data while the probabilistic encoder is trained with on-policy data. In our experimental evaluation, we demonstrate 20-100x improvement in meta-training sample efficiency over prior methods on six simulated continuous control meta-learning domains.

While the data efficiency of PEARL is sufficient to perform meta-training on a real robotic system, challenges remain to apply meta-RL in the real world. Applying these algorithms to real-world robotic systems requires handling the raw sensory observations collected by a robot's on-board sensors. In principle, deep reinforcement learning (RL) algorithms can directly map sensory inputs to actions. However, this automation comes at a steep cost in sample efficiency since the agent must learn to interpret observations from reward supervision alone. Fortunately, unsupervised learning of general-purpose latent state (or dynamics) models can serve as an additional training signal to help solve the representation learning problem (Finn et al., 2016; Ghadirzadeh et al., 2017; A. Lee et al., 2019; M. Zhang et al., 2019). In Section 5.3, we seek to leverage the benefits of latent state models for representation learning to design the meta-RL algorithm MELD that can acquire new skills quickly in the real world. Our key insight is that the same latent dynamics models that greatly improve efficiency in end-to-end single-task RL can *also*, with minimal modification, be used for meta-RL by treating the unknown task information as part of the latent state estimated from experience. The trained system quickly learns a new task by inferring the posterior belief over the hidden variable and executing the conditional meta-learned policy. We find that MELD substantially outperforms prior work on several vision-based simulated domains, and then demonstrate that MELD can perform Ethernet cable insertion into ports at novel locations and orientations using a real WidowX.

5.1 RELATED WORK

Meta-learning. Our work builds on the meta-learning framework (Schmidhuber, 1987; Bengio et al., 1990; Thrun and Pratt, 1998) in the context of reinforcement learning.

Recently, meta-RL methods have been developed for meta-learning dynamics models (Nagabandi et al., 2019; Sæmundsson et al., 2018; Doshi-Velez and Konidaris, 2016) and policies (Finn et al., 2017a; Duan et al., 2016; Mishra et al., 2018) that can quickly adapt to new tasks.

Recurrent (Duan et al., 2016; Wang and Hebert, 2016) and recursive (Mishra et al., 2018) meta-RL methods adapt to new tasks by aggregating experience into a latent representation on which the policy is conditioned. These approaches can be categorized into what we will call *context-based* meta-RL methods, since a neural network is trained to take experience as input as a form of task-specific context. Similarly, our approach can also be considered context-based; however, we represent task contexts with probabilistic latent variables, enabling reasoning over task uncertainty. In the PEARL algorithm, instead of using recurrence, we leverage the Markov property in our permutation-invariant encoder to aggregate experience, enabling fast optimization especially for long-horizon tasks while mitigating overfitting. While prior work has studied methods that can train recurrent Q-functions with off-policy Q-learning methods, such methods have often been applied to much simpler tasks (Heess et al., 2015), and in discrete environments (Hausknecht and Stone, 2015). Indeed, our own experiments in Section 5.2.3 demonstrate that straightforward incorporation of recurrent policies with off-policy learning is difficult. Contextual methods have also been applied to imitation learning by conditioning the policy on a learned embedding of a demonstration and optimizing with behavior cloning (Duan et al., 2017; James et al., 2018).

In contrast to context-based methods, *gradient-based* meta-RL methods learn from aggregated experience using policy gradients (Finn et al., 2017a; Stadie et al., 2018; Rothfuss et al., 2018; T. Xu et al., 2018; Houthooft et al., 2018; Mendonca et al., 2019), meta-learned loss functions (Sung et al., 2017; Houthooft et al., 2018), or hyperparameters (Z. Xu et al., 2018). These methods focus on on-policy meta-learning. We instead focus on meta-learning from off-policy data, which is non-trivial to do with methods based on policy gradients and evolutionary optimization algorithms. Beyond substantial sample efficiency improvements, we also empirically find that our methods PEARL and MELD are able to reach higher asymptotic performance, in comparison to methods using policy gradients.

Outside of RL, meta-learning methods for few-shot supervised learning problems have explored a wide variety of approaches and architectures (Santoro et al., 2016; Vinyals et al., 2016; Ravi and Larochelle, 2017; Oreshkin et al., 2018). The permutation-invariant embedding function used in PEARL is inspired by the embedding function of prototypical networks (Snell et al., 2017). While they use a distance metric in a learned, deterministic

embedding space to classify new inputs, our embedding is probabilistic and is used to condition the behavior of an RL agent.

Probabilistic meta-learning. Prior work has applied probabilistic models to meta-learning in both supervised and RL domains. Hierarchical Bayesian models have been used to model few-shot learning (Fei-Fei et al., 2003; Tenenbaum, 1999), including approaches that perform gradient-based adaptation (Grant et al., 2018; J. Yoon et al., 2018). For supervised learning, Rusu et al. (2019); J. Gordon et al. (2019); Finn et al. (2018) adapt model predictions using probabilistic latent task variables inferred via amortized approximate inference. We extend this idea to off-policy meta-RL. In the context of multi-task RL, Hausman et al. (2018) conditions the policy on inferred task variables, but the aim is to compose tasks via the embedding space, while we focus on rapid adaptation to new tasks. Similar to PEARL, MAESN (Gupta et al., 2018b) uses structured noise to induce temporally-extended exploration strategies; however in PEARL strategy manifests as posterior sampling. In prior work in RL, posterior sampling (Strens, 2000; Osband et al., 2013) maintains a posterior over possible MDPs and enables temporally extended exploration by acting optimally according to a sampled MDP. PEARL performs a meta-learned variant of this method: the probabilistic context captures the current uncertainty over the task; sampling it allows the agent to explore in new tasks in a similarly structured manner.

Partially observed MDPs. Adaptation at test time in meta-RL can be viewed as a special case of RL in a POMDP (Kaelbling et al., 1998) by including the task as the unobserved part of the state. We use a variational approach related to Igl et al. (2018) to estimate belief over the task. While they focus on solving general POMDPs, in the PEARL algorithm we leverage the additional structure imposed by the meta-learning problem to simplify inference, and use posterior sampling for exploration in new tasks. Other works published after PEARL have also formalized meta-RL as a special kind of POMDP in which the hidden state is constant throughout a task (Zintgraf et al., 2019; Humplik et al., 2019; Perez et al., 2020). In MELD, we challenge this idea that meta-RL algorithms should leverage the convention that tasks remain constant throughout episodes, demonstrating that meta-RL can be performed by general latent state estimation algorithms. The MELD model estimates a time-varying hidden state that captures both state and task information, rendering the same algorithm applicable to problems with both stationary and non-stationary sources of uncertainty.

Latent State Inference in RL. A significant challenge in real-world robotic learning is contending with the complex, high-dimensional, and noisy observations from the robot's sensors. To handle general partial observability, recurrent policies can persist information

over longer time horizons (Yadaiah and Sowmya, 2006; Heess et al., 2015; Hausknecht and Stone, 2015), while explicit state estimation approaches maintain a probabilistic belief over the current state of the agent and update it given experience (Kaelbling et al., 1998; Pineau et al., 2003; Ross et al., 2011; Deisenroth and Peters, 2012; Karkus et al., 2017; Igl et al., 2018; Gregor and Besse, 2018). In our experiments we focus on learning from image observations, which presents a state representation learning challenge that has been studied in detail. End-to-end deep RL algorithms can learn state representations implicitly, but currently suffer from poor sample efficiency due to the added burden of representation learning (Mnih et al., 2013; Levine et al., 2016; Singh et al., 2020). Pre-trained state estimation systems can predict potentially useful features such as object locations and pose (Tremblay et al., 2018; Visak Kumar et al., 2019); however, these approaches require ground truth supervision. On the other hand, unsupervised learning techniques can improve sample efficiency without access to additional supervision (Lange et al., 2012; Finn et al., 2016; Schmidt et al., 2016; Ghadirzadeh et al., 2017; Florence et al., 2019; Yarats et al., 2019; Sax et al., 2019). Latent dynamics models capture the time-dependence of observations and provide a learned latent space in which RL can be tractably performed (Watter et al., 2015; Karl et al., 2016; M. Zhang et al., 2019; Hafner et al., 2019; Gelada et al., 2019; A. Lee et al., 2019). In MELD, we generalize the learned latent variable to encode not only the state but also the task at hand, enabling efficient meta-RL from images.

RL and Meta-RL for Robotics. While prior work has obtained good results with geometric and force control approaches for a wide range of manipulation tasks (Bicchi and Vijay Kumar, 2000; Pereira et al., 2004; Henrich and Wörn, 2012), including insertion tasks (Kronander et al., 2014; Newman et al., 2001) such as those in our evaluation, such approaches typically require considerable manual design effort for each task. RL algorithms offer an automated alternative that has been demonstrated on a variety of robotic tasks (Kober et al., 2013) including insertion (Gullapalli et al., 1994; Levine et al., 2016; Zeng et al., 2018; M. A. Lee et al., 2018; Schoettler et al., 2019). Although these policies learn impressive skills, they typically do not transfer to other tasks and must be re-trained from scratch for each task.

Meta-learning approaches that enable few-shot adaptation have been studied with real systems for imitation learning (Finn et al., 2017b; Yu et al., 2018; James et al., 2018; Bonardi et al., 2019) and goal inference (Xie et al., 2018), but direct meta-RL in the real world has received comparatively little attention. Adapting to different environment parameters has been explored in the sim2real setting for table-top hockey (Arndt et al., 2019) and legged locomotion (X. Song et al., 2020), and in the model-based RL setting

for millirobot locomotion (Nagabandi et al., 2018). In Section 5.3.3, we demonstrate that our algorithm MELD can perform meta-RL trained from images in the real world.

5.2 PEARL: PROBABILISTIC EMBEDDINGS FOR ACTOR-CRITIC META-RL

To achieve both meta-training efficiency and rapid adaptation to new tasks, we propose an approach that integrates online inference of probabilistic context variables with existing off-policy RL algorithms. Rapid adaptation requires reasoning about distributions: when exposed to a new task for the first time, the optimal meta-learned policy must carry out a stochastic exploration procedure to visit potentially rewarding states, as well as adapt to the task at hand (Gupta et al., 2018b). During meta-training, we learn a probabilistic encoder that accumulates the necessary statistics from past experience into the context variables that enable the policy to perform the task. At meta-test time, when the agent is faced with an unseen task, the context variables can be sampled and held constant for the duration of an episode, enabling temporally-extended exploration. The collected trajectories are used to update the posterior over the context variables, achieving rapid trajectory-level adaptation. In effect, our method adapts by sampling "task hypotheses," attempting those tasks, and then evaluating whether the hypotheses were correct or not. Disentangling task inference from action serves to minimize distribution mismatch between meta-train and meta-test; the policy can be optimized with off-policy data while the probabilistic encoder is trained with on-policy data.

We call this algorithm Probabilistic Embeddings for Actor-critic meta-RL (PEARL). Our method achieves excellent sample efficiency during meta-training, enables fast adaptation by accumulating experience online, and performs structured exploration by reasoning about uncertainty over tasks. In our experimental evaluation, we demonstrate state-of-the-art results with 20-100x improvement in meta-training sample efficiency and substantial increases in asymptotic performance over prior state-of-the-art on six continuous control meta-learning domains. We examine how PEARL conducts structured exploration to adapt rapidly to new tasks in a 2-D navigation environment with sparse rewards. Our open-source implementation of PEARL can be found at `https://github.com/katerakelly/oyster`.

5.2.1 *Probabilistic Latent Context*

We capture knowledge about how the current task should be performed in a latent probabilistic context variable Z, on which we condition the policy as $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{z})$ in order to

adapt its behavior to the task. Meta-training consists of leveraging data from a variety of training tasks to learn to infer the value of Z from a recent history of experience in the new task, as well as optimizing the policy to solve the task given samples from the posterior over Z. In this section we describe the structure of the meta-trained inference mechanism. We address how meta-training can be performed with off-policy RL algorithms in Section 5.2.2.

Definitions See Chapter 3 for preliminaries on meta-learning and reinforcement learning. We briefly recap the meta-RL problem statement here. Formally, a task $\mathcal{T} = \{p(\mathbf{s}_0), p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t), r(\mathbf{s}_t, \mathbf{a}_t)\}$ consists of an initial state distribution $p(\mathbf{s}_0)$, transition distribution $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, and reward function $r(\mathbf{s}_t, \mathbf{a}_t)$. Given a set of training tasks sampled from task distribution $p(\mathcal{T})$, the meta-training process learns a policy that adapts to the task at hand by conditioning on the history of past transitions, which we refer to as *context* $\mathbf{c}$. Let $\mathbf{c}_n^{\mathcal{T}} = (\mathbf{s}_n, \mathbf{a}_n, r_n, \mathbf{s}'_n)$ be one transition in the task $\mathcal{T}$ so that $\mathbf{c}_{1:N}^{\mathcal{T}}$ comprises the experience collected so far. At test-time, the policy must adapt to a new task drawn from $p(\mathcal{T})$.

Modeling and Learning Latent Contexts To enable adaptation, the latent context Z must encode salient information about the task. Recall that $\mathbf{c}_{1:N}^{\mathcal{T}}$ comprises experience collected so far; throughout this section we will often write $\mathbf{c}$ for simplicity. We adopt an amortized variational inference approach (Kingma and Welling, 2014; Rezende et al., 2014; Alemi et al., 2016) to learn to infer Z. We train an *inference network* $q_\phi(\mathbf{z}|\mathbf{c})$, parameterized by ϕ, that estimates the posterior $p(\mathbf{z}|\mathbf{c})$. In a generative approach, this can be achieved by optimizing $q_\phi(\mathbf{z}|\mathbf{c})$ to reconstruct the MDP by learning a predictive models of reward and dynamics. Alternatively, $q_\phi(\mathbf{z}|\mathbf{c})$ can be optimized in a model-free manner to model the state-action value functions or to maximize returns through the policy over the distribution of tasks. Assuming this objective to be a log-likelihood, the resulting variational lower bound is:

$$\mathbb{E}_{\mathcal{T}}[\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{c}^{\mathcal{T}})}[R(\mathcal{T}, \mathbf{z}) + \beta D_{KL}(q_\phi(\mathbf{z}|\mathbf{c}^{\mathcal{T}})||p(\mathbf{z}))]] \tag{8}$$

where $p(\mathbf{z})$ is a unit Gaussian prior over Z and $R(\mathcal{T}, \mathbf{z})$ could be a variety of objectives, as discussed above. The KL divergence term can also be interpreted as the result of a variational approximation to an information bottleneck (Alemi et al., 2016) that constrains the mutual information between Z and $\mathbf{c}$. Intuitively, this bottleneck constrains $\mathbf{z}$ to contain only information from the context that is necessary to adapt to the task at hand, mitigating overfitting to training tasks. The parameters of q_ϕ are optimized during meta-training and then fixed; at meta-test time the latent context for a new task is simply inferred from gathered experience.

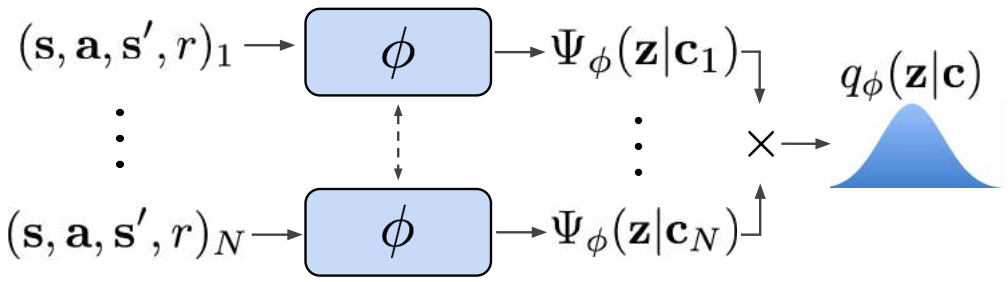


Figure 12: **Inference network architecture**. The amortized inference network predicts the posterior over the latent context variables $q_\phi(\mathbf{z}|\mathbf{c})$ as a permutation-invariant function of prior experience.

In designing the architecture of the inference network $q_\phi(\mathbf{z}|\mathbf{c})$, we would like it to be expressive enough to capture minimal sufficient statistics of task-relevant information, without modeling irrelevant dependencies. We note that an encoding of a fully observed MDP can be permutation-invariant with respect to sampled transitions that consist of $\{\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i\}$. The transition and reward functions (and thus the MDP) can be reconstructed from an unordered set of such transitions. It follows that a collection of such transitions is sufficient to train a value function or infer what the task is. With this observation in mind, we choose a permutation-invariant representation for $q_\phi(\mathbf{z}|\mathbf{c}_{1:N})$, modeling it as a product of independent factors

$$q_\phi(\mathbf{z}|\mathbf{c}_{1:N}) \propto \Pi_{n=1}^{N} \Psi_\phi(\mathbf{z}|\mathbf{c}_n) \tag{9}$$

To keep the method tractable, we use Gaussian factors $\Psi_\phi(\mathbf{z}|\mathbf{c}_n) = \mathcal{N}(f_\phi^\mu(\mathbf{c}_n), f_\phi^\sigma(\mathbf{c}_n))$, which result in a Gaussian posterior. The function f_ϕ, represented as a neural network parameterized by ϕ, predicts the mean μ as well as the variance σ as a function of the $\mathbf{c}_n$, is shown in Figure 12.

Posterior Sampling and Exploration via Latent Contexts Modeling the latent context as probabilistic allows us to make use of posterior sampling for efficient exploration at meta-test time. Posterior sampling (Strens, 2000; Osband et al., 2013) for exploration in RL begins with a prior distribution over MDPs, computes a posterior distribution conditioned on the experience it has seen so far, and executes the optimal policy for a sampled MDP for the duration of an episode as an efficient method for exploration. In particular, acting optimally according to a random MDP allows for temporally extended exploration, meaning that the agent can act to test hypotheses even when the results of actions are not immediately informative of the task.

In the single-task deep RL setting, posterior sampling and the benefits of deep ex-

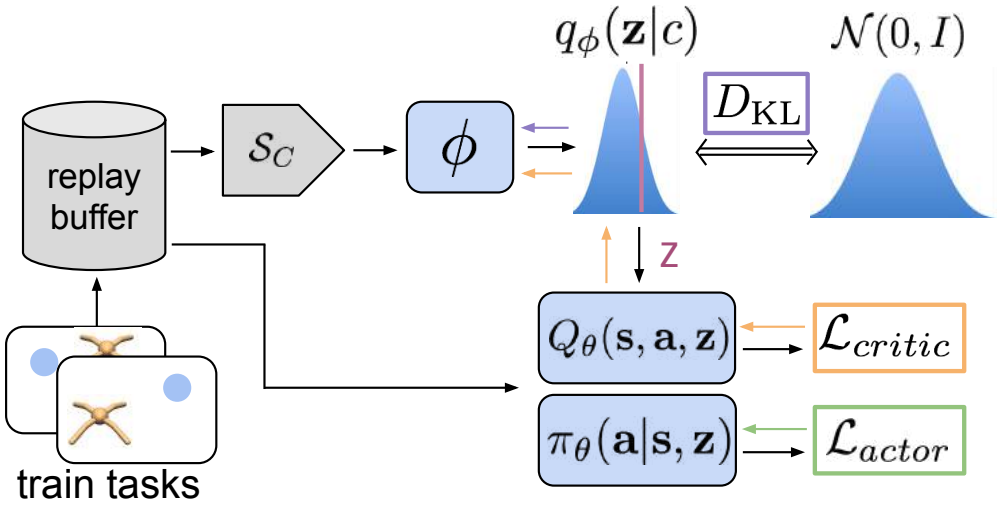


Figure 13: **Meta-training procedure.** The inference network q_ϕ uses context data to infer the posterior over the latent context variable Z, which is passed as an input to the actor and critic. q_ϕ is optimized with gradients from the critic as well as from an information bottleneck on Z. De-coupling the data sampling strategies for context ($\mathcal{S}_C$) and actor-critic batches is important for off-policy learning, see ablation in Section 5.2.3.

ploration has been explored by Osband et al. (2016), which maintains an approximate posterior over value functions via bootstrapping. In contrast, our method PEARL directly infers a posterior over the latent context Z, which may encode the MDP itself if optimized for reconstruction, optimal behaviors if optimized for the policy, or the value function if optimized for a critic. Our meta-training procedure leverages training tasks to learn a prior over Z that captures the distribution over tasks and also learns to efficiently use experience to infer new tasks. At meta-test time, we initially sample $\mathbf{z}$'s from the prior and execute according to each $\mathbf{z}$ for an episode, thus exploring in a temporally extended and diverse manner. We then use the collected experience to update theposterior and continue exploring coherently in a manner that acts more and more optimally as our belief narrows, akin to posterior sampling.

5.2.2 *Off-Policy Meta-Reinforcement Learning*

While our probabilistic context model is straightforward to combine with on-policy policy gradient methods, a primary goal of our work is to enable efficient off-policy meta-reinforcement learning, where the number of samples for *both* meta-training and fast adaptation is minimal. The efficiency of the meta-training process is largely disregarded

in prior work, which make use of stable but relatively inefficient on-policy algorithms (Duan et al., 2016; Finn et al., 2017a; Gupta et al., 2018b; Mishra et al., 2018). However, designing off-policy meta-RL algorithms is non-trivial partly because modern meta-learning is predicated on the assumption that the distribution of data used for adaptation will match across meta-training and meta-test. In RL, this implies that since at meta-test time on-policy data will be used to adapt, on-policy data should be used during meta-training as well. Furthermore, meta-RL requires the policy to reason about *distributions*, so as to learn effective stochastic exploration strategies. This problem inherently cannot be solved by off-policy RL methods that minimize temporal-difference error, as they do not have the ability to directly optimize for distributions of states visited. In contrast, policy gradient methods have direct control over the actions taken by the policy. Given these two challenges, a naive combination of meta-learning and value-based RL could be ineffective. In practice, we were unable to optimize such a method.

As detailed in Section 5.2.1, we address the issue of meta-learning exploration strategies off-policy by modeling the latent context variable as probabilistic, enabling exploration via posterior sampling. We address the distribution shift between off-policy meta-training data and on-policy test-time adaptation data by noting that the data used to train the encoder need not be the same as the data used to train the policy. The policy can treat the context $\mathbf{z}$ as part of the state in an off-policy RL loop, while the stochasticity of the exploration process is provided by the uncertainty in the encoder $q(\mathbf{z}|\mathbf{c})$. The actor and critic are always trained with off-policy data sampled from the entire replay buffer $\mathcal{B}$. We define a sampler $\mathcal{S}_{\mathbf{c}}$ to sample context batches for training the encoder. Allowing $\mathcal{S}_{\mathbf{c}}$ to sample from the entire buffer presents too extreme of a distribution mismatch with on-policy test data and empirical performance suffers, see Section 5.2.3. However, the context does not need to be strictly on-policy; we find that an in-between strategy of sampling from a replay buffer of recently collected data retains on-policy performance with better efficiency. We summarize our training procedure in Figure 13 and Algorithm 3. Meta-testing is described in Algorithm 4.

Implementation We build our algorithm on top of the soft actor-critic algorithm (SAC) (Haarnoja et al., 2018), an off-policy actor-critic method based on the maximum entropy RL objective which augments the traditional sum of discounted returns with the entropy of the policy. SAC exhibits good sample efficiency and stability. We optimize the parameters of the inference network $q(\mathbf{z}|\mathbf{c})$ jointly with the parameters of the actor $\pi_\theta(\mathbf{a}|\mathbf{s},\mathbf{z})$ and critic $Q_\theta(\mathbf{s},\mathbf{a},\mathbf{z})$, using the reparameterization trick (Kingma and Welling, 2014) to compute gradients for parameters of $q_\phi(\mathbf{z}|\mathbf{c})$ through sampled $\mathbf{z}$'s. We train the inference network using gradients from the Bellman update for the critic. We found em-

pirically that training the encoder to recover the state-action value function outperforms optimizing it to maximize actor returns, or reconstruct states and rewards. The critic loss can then be written as,

$$\mathcal{L}_{\text{critic}} = \mathbb{E}_{\substack{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\sim\mathcal{B} \\ \mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{c})}}[Q_\theta(\mathbf{s},\mathbf{a},\mathbf{z}) - (r + \bar{V}(\mathbf{s}',\bar{\mathbf{z}}))]^2 \tag{10}$$

where $\bar{V}$ is a target network and $\bar{\mathbf{z}}$ indicates that gradients are not being computed through it. The actor loss is nearly identical to SAC, with the additional dependence on $\mathbf{z}$ as a policy input.

$$\mathcal{L}_{\text{actor}} = \mathbb{E}_{\substack{\mathbf{s}\sim\mathcal{B},\mathbf{a}\sim\pi_\theta \\ \mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{c})}}\left[D_{\text{KL}}\left(\pi_\theta(\mathbf{a}|\mathbf{s},\bar{\mathbf{z}})\middle\|\frac{\exp(Q_\theta(\mathbf{s},\mathbf{a},\bar{\mathbf{z}}))}{\mathcal{Z}_\theta(\mathbf{s})}\right)\right] \tag{11}$$

Note that the context used to infer $q_\phi(\mathbf{z}|\mathbf{c})$ is distinct from the data used to construct the actor and critic losses. As described above, during meta-training we sample context batches separately from actor-critic batches. Concretely, the context sampler $\mathcal{S}_\mathbf{c}$ samples uniformly from the most recently collected batch of data, recollected every 1000 meta-training optimization steps. The actor and critic are trained with batches of transitions drawn uniformly from the entire replay buffer.

Algorithm 1 PEARL Meta-training

Require: Batch of training tasks $\{\mathcal{T}_i\}_{i=1...T}$ from $p(\mathcal{T})$, learning rates $\alpha_1, \alpha_2, \alpha_3$

Initialize replay buffers $\mathcal{B}^i$ for each training task
while not done **do**
  for each $\mathcal{T}_i$ **do**
    Initialize context $\mathbf{c}^i = \{\}$
    for $k = 1, \ldots, K$ **do**
      Sample $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{c}^i)$
      Gather data from $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{z})$ and add to $\mathcal{B}^i$
      Update $\mathbf{c}^i = \{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j)\}_{j:1...N} \sim \mathcal{B}^i$
    end for
  end for
  for step in training steps **do**
    for each $\mathcal{T}_i$ **do**
      Sample context $\mathbf{c}^i \sim \mathcal{S}_c(\mathcal{B}^i)$ and RL batch $b^i \sim \mathcal{B}^i$
      Sample $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{c}^i)$
      $\mathcal{L}^i_{\text{actor}} = \mathcal{L}_{\text{actor}}(b^i, \mathbf{z})$
      $\mathcal{L}^i_{\text{critic}} = \mathcal{L}_{\text{critic}}(b^i, \mathbf{z})$
      $\mathcal{L}^i_{\text{KL}} = \beta D_{\text{KL}}(q(\mathbf{z}|\mathbf{c}^i)||r(\mathbf{z}))$
    end for
    $\phi \leftarrow \phi - \alpha_1 \nabla_\phi \sum_i \left(\mathcal{L}^i_{\text{critic}} + \mathcal{L}^i_{\text{KL}}\right)$
    $\theta_\pi \leftarrow \theta_\pi - \alpha_2 \nabla_\theta \sum_i \mathcal{L}^i_{\text{actor}}$
    $\theta_Q \leftarrow \theta_Q - \alpha_3 \nabla_\theta \sum_i \mathcal{L}^i_{\text{critic}}$
  end for
end while

Algorithm 2 PEARL Meta-testing

Require: test task $\mathcal{T} \sim p(\mathcal{T})$

Initialize context $\mathbf{c}^{\mathcal{T}} = \{\}$
for $k = 1, \ldots, K$ **do**
  Sample $z \sim q_\phi(\mathbf{z}|c^{\mathcal{T}})$
  Roll out policy $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{z})$ to collect data $D^{\mathcal{T}}_k = \{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j)\}_{j:1...N}$
  Accumulate context $\mathbf{c}^{\mathcal{T}} = \mathbf{c}^{\mathcal{T}} \cup D^{\mathcal{T}}_k$
end for

5.2.3 *Experiments*

In our experiments, we assess the performance of our method and analyze its properties. We first evaluate how our approach compares to prior meta-RL methods, especially in terms of sample efficiency, on several benchmark meta-RL problems. We then examine how probabilistic context and posterior sampling enable rapid adaptation via structured exploration strategies in sparse reward settings. Finally, we evaluate the specific design choices in our algorithm through ablations.

Sample Efficiency and Performance We evaluate PEARL on six continuous control environments focused around robotic locomotion, simulated via the MuJoCo simulator (Todorov et al., 2012). These locomotion task families require adaptation across reward functions (walking direction for Half-Cheetah-Fwd-Back, Ant-Fwd-Back, Humanoid-Direc-2D, target velocity for Half-Cheetah-Vel, and goal location for Ant-Goal-2D) or across dynamics (random system parameters for Walker-2D-Params). These meta-RL benchmarks were previously introduced by Finn et al. (2017a) and Rothfuss et al. (2018). All tasks have horizon length 200. We compare to existing policy gradient meta-RL methods ProMP (Rothfuss et al., 2018) and MAML-TRPO (Finn et al., 2017a) using publicly available code. We also re-implement the recurrence-based policy gradient RL^2 method (Duan et al., 2016) with PPO (Schulman et al., 2017). The results of each algorithm are averaged across three random seeds. We attempted to adapt recurrent DDPG (Heess et al., 2015) to our setting, but were unable to obtain reasonable results with this method. We hypothesize that this is due to a combination of factors including the distribution mismatch in the adaptation data discussed in Section 5.2.2 and the difficulty of training with trajectories rather than decorrelated transitions. This approach does not explicitly infer a belief over the task as we do, instead leaving the burden of both task inference and optimal behavior to the RNN. In PEARL, decoupling task inference from the policy allows us the freedom to choose the encoder data and objective that work best with off-policy learning. We experiment with recurrent architectures in the context of our own method in the ablations.

To evaluate on the meta-testing tasks, we perform adaptation at the trajectory level, where the first trajectory is collected with context variable $\mathbf{z}$ sampled from the prior $p(\mathbf{z})$. Subsequent trajectories are collected with $\mathbf{z} \sim q(\mathbf{z}|\mathbf{c})$ where the context is aggregated over all trajectories collected. To compute final test-time performance, we report the average returns of trajectories collected after two trajectories have been aggregated into the context. Notably, we find RL^2 performs much better on these benchmarks than previously reported, possibly due to using PPO for optimization and selecting better hyper-

parameters. We observe that PEARL significantly outperforms prior meta-RL methods across all domains in terms of both asymptotic performance and sample efficiency, see Figure 14. We truncate the x-axis at the number of timesteps required for PEARL to converge; see Appendix A for the full timescale version. We find that PEARL uses 20-100x fewer samples during meta-training than previous meta-RL approaches while improving final asymptotic performance by 50-100% in five of the six domains.

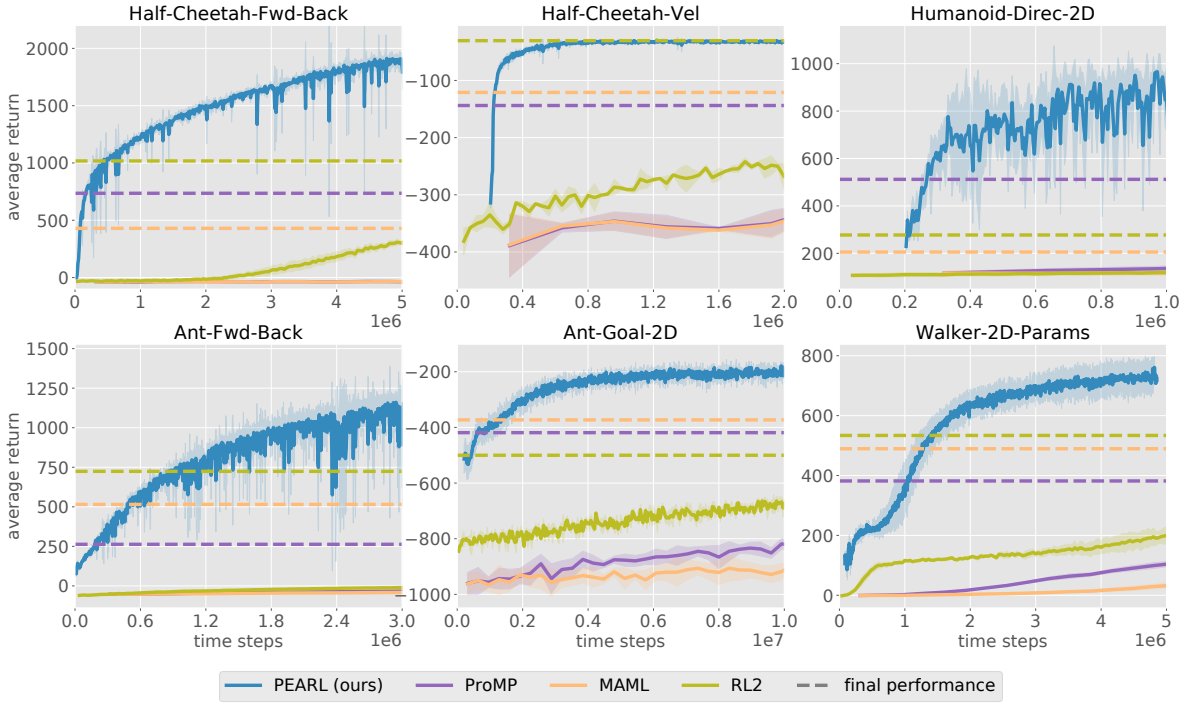


Figure 14: **Meta-learning continuous control**. Test-task performance vs. samples collected during *meta-training*. Our approach PEARL outperforms previous meta-RL methods both in terms of asymptotic performance and meta-training sample efficiency across six benchmark tasks. Dashed lines correspond to the maximum return achieved by each baseline after 1e8 steps. By leveraging off-policy data during meta-training, PEARL is 20-100x more sample efficient than the baselines, and achieves consistently better or equal final performance compared to the best performing prior method in each environment. See Appendix A for the full timescale version of this plot.

Posterior Sampling For Exploration In this section we evaluate whether posterior sampling in our model enables effective exploration strategies in sparse reward MDPs. Intuitively, by sampling from the prior context distribution $p(\mathbf{z})$, the agent samples a hypothesis according to the distribution of training tasks it has seen before. As the agent acts in the environment, the context posterior $p(\mathbf{z}|\mathbf{c})$ is updated, allowing it to reason over multiple hypotheses to determine the task. We demonstrate this behavior with a 2-D navigation task in which a point robot must navigate to different goal locations on edge of a semi-circle. We sample 100 random goals from the distribution for training and 20 for testing. A reward is given only when the agent is within a certain radius of the goal. We experiment with radius 0.2 and 0.8. The horizon length is 20 steps. While our aim is to adapt to new tasks with sparse rewards, meta-training with sparse rewards is extremely difficult as it amounts to solving many sparse reward tasks from scratch. For simplicity we therefore assume access to the dense reward during meta-training, as done by Gupta et al. (2018b), but this burden could also be mitigated with task-agnostic exploration strategies. In this setting, we compare to MAESN (Gupta et al., 2018b), a prior method that also models probabilistic task variables and performs on-policy gradient-based meta-learning. We demonstrate we are able to adapt to the new sparse goal in fewer trajectories. Even with fewer samples, PEARL also outperforms MAESN in terms of final performance. In Figure 15 we compare adaptation performance on test tasks. In addition to achieving higher returns and adapting

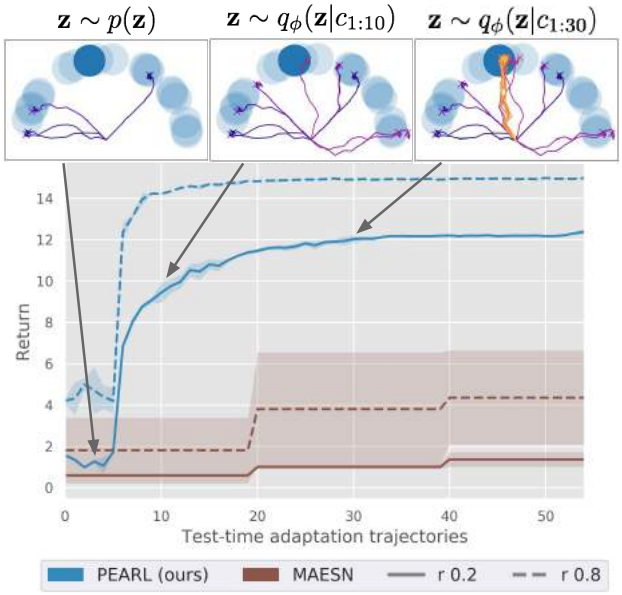


Figure 15: **Sparse 2D navigation**. The agent must navigate to a previously unseen goal (dark blue, other test goals in light blue) with reward given only when inside the goal radius – radius of 0.2 (illustrated) and 0.8 are tested here. The agent is trained to navigate to a training set of goals, then tested on a distinct set of unseen test goals. By using posterior sampling to explore efficiently, PEARL is able to start adapting to the task after collecting on average only 5 trajectories, outperforming MAESN (Gupta et al., 2018b).

faster, PEARL is also more efficient during meta-training. Our results were achieved with 100x fewer meta-training timesteps than MAESN.

Ablations In this section we ablate the components of our approach. We examine our choice of permutation-invariant encoder for the latent context Z by comparing it to a conventional choice for encoding MDPs, a recurrent network (Duan et al., 2016; Heess et al., 2015). Note that while in the benchmark experiments we considered a recurrent-based baseline similar to recurrent DDPG (Heess et al., 2015), here we retain all other features of our method and ablate only the encoder structure. We backprop through the RNN to 100 timesteps. We sample the context as full trajectories rather than unordered transitions as in PEARL. We experiment with two ways of sampling the actor-critic batch: (1) unordered transitions, as in PEARL ("RNN de-correlated"), and (2) sets of trajectories ("RNN correlated"). In Figure 16, we compare the test task performance in Half-Cheetah-Vel as a function of the number of meta-training samples. Replacing our encoder with an RNN results in comparable performance to PEARL, at the cost of slower optimization. However, sampling trajectories for the actor-critic batch results in a steep drop in performance. This result demonstrates the importance of de-correlating the samples used for the RL objective.

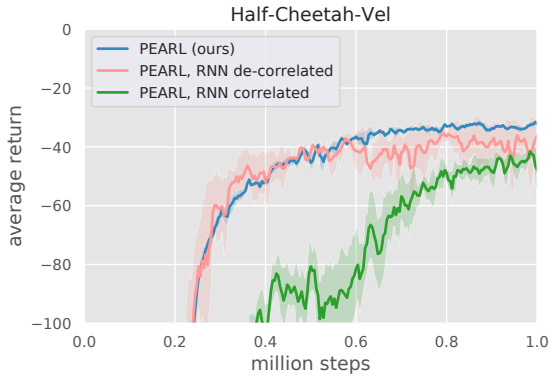


Figure 16: **Recurrent encoder ablation**. We compare our encoder to a recurrent network. For the RNN, we sample context as trajectories rather than unordered transitions. Sampling the actor-critic batch as de-correlated transitions ("RNN de-correlated") fares much better than sampling trajectories ("RNN correlated").

Next, we ablate the context sampling strategy used during training. With sampler $\mathcal{S}_c$, PEARL samples batches of unordered transitions that are **(1)** restricted to samples recently collected by the policy, and **(2)** distinct from the set of transitions collected by the RL sampler. We consider two other options for $\mathcal{S}_c$:

- sample fully off-policy data from the entire replay buffer, but distinct from the actor-critic batch ("off-policy")
- use the same off-policy actor-critic batch as the context batch ("off-policy same batch")

Results are shown in Figure 17. Sampling context off-policy significantly hurts performance. Using the same batch to train the actor-critic and encoder in this case helps, perhaps because the correlation makes learning easier. Overall these results demonstrate the importance of careful data sampling in off-policy meta-RL.

Finally, we examine the importance of modeling the latent context as probabilistic. As discussed in Section 5.2.1, we hypothesize that a probabilistic context is particularly important in sparse reward settings because it allows the agent to model a distribution over tasks and conduct exploration via posterior sampling. To test this empirically, we train a deterministic version of PEARL by reducing the distribution $q_\phi(\mathbf{z}|\mathbf{c})$ to a point estimate, and compare to PEARL on the sparse 2D navigation domain in Figure 18. With deterministic latent context, the only stochasticity comes from the policy and is thus time-invariant, hindering temporally extended exploration and preventing the agent from solving the task.

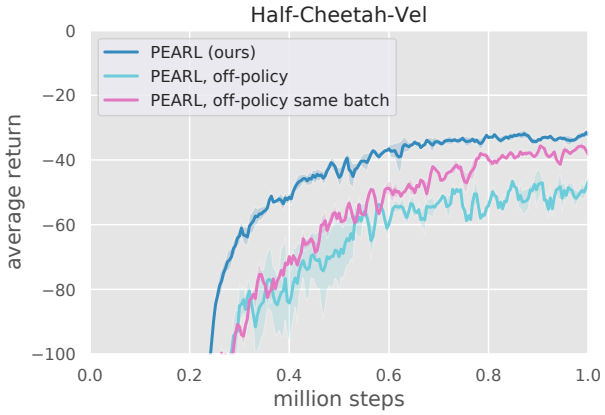


Figure 17: **Context sampling ablation**. To train the encoder, PEARL samples recently collected transitions de-correlated with the actor-critic batches. We compare to sampling context from the entire history ("off-policy context"), and using the actor-critic batch as the context ("off-policy same batch").

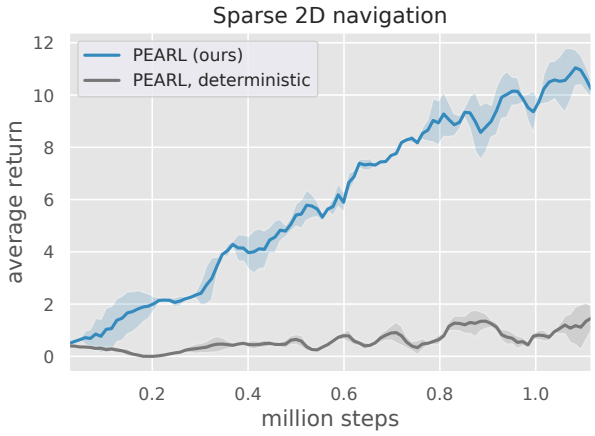


Figure 18: **Deterministic latent context**. We compare PEARL to a variant with deterministic latent context on the sparse reward 2D navigation domain. Without a mechanism to reason about uncertainty over tasks, this approach is unable to explore effectively.

5.2.4 *Discussion*

In this section we proposed a novel meta-RL algorithm, PEARL, which adapts to new tasks by performing inference over a latent context variable on which the policy is conditioned. Our approach is particularly amenable to using off-policy RL al-

gorithms for meta-training as it decouples the problems of inferring the task and solving it, allowing for off-policy meta-training while minimizing mismatch between train and test context distributions. Modeling the latent context as probabilistic enables posterior sampling for exploration at test time, resulting in temporally extended exploration behaviors that enhance adaptation efficiency. Our approach achieves superior performance compared to prior meta-RL algorithms while requiring 20-100x fewer samples on a diverse set of continuous control meta-RL domains.

Based on these results of meta-learning with simulated robots, we can make a rough estimate of the training time required on a real robotic system. Assuming the control frequency is 30Hz and that data collection runs continuously on a single robot, the baselines would require 1-2 months of interaction for meta-training, while PEARL might only need 24-48 hours. Motivated by this result, in the next section we turn to how we can enable real robots to quickly acquire new skills via meta-learning.

5.3 MELD: LATENT STATE MODELS FOR META-RL FROM IMAGES

While the sample efficiency of PEARL is sufficient for running on a real-world robotic system, in addition to data collection constraints, robots additionally must contend with the high-dimensional observations from their sensors. In single-task RL, latent state models improve the efficiency of RL by leveraging unsupervised learning to learn a compact latent state representation from a history of observations. Our insight is that these same models can *also*, with minimal modification, be used for meta-RL by treating the unknown task information as part of the latent state to be estimated from experience. In this section we formalize the connection between latent state inference and meta-RL, and leverage this insight in our proposed algorithm MELD, Meta-RL with Latent Dynamics. To derive MELD, we cast meta-RL and latent state inference into a single partially observed Markov decision process (POMDP) in which task and state variables are aspects of a more general per-time step hidden variable. Concretely, we represent the agent's belief over the hidden variable as the variational posterior in a sequential VAE latent state model that takes observations and rewards as input, and we condition the agent's policy on this belief. During meta-training, the latent state model and policy are trained across a fixed set of training tasks sampled from the task distribution. The trained system can then quickly learn a new task from the distribution by inferring the posterior belief over the hidden variable and executing the conditional meta-learned policy.

We find in simulation that MELD substantially outperforms prior work on several challenging locomotion and manipulation problems, such as running at varying velocities,

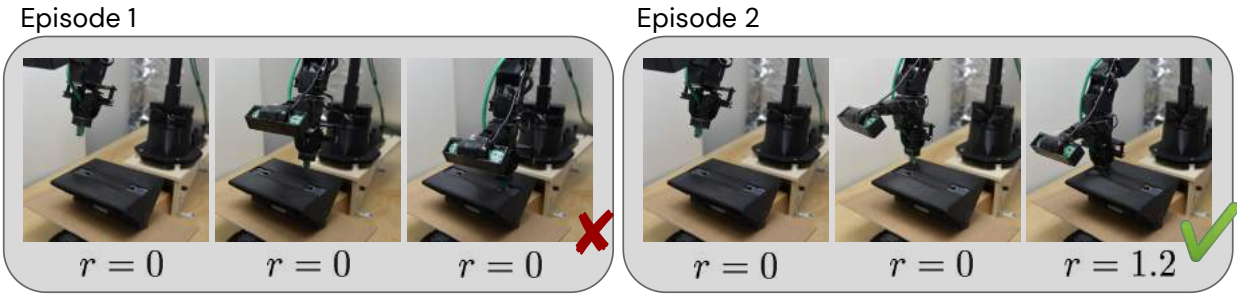


Figure 19: At test time our algorithm MELD enables a 5-DoF WidowX robot to insert the ethernet cable into a novel insertion location and orientation within two episodes of experience, operating from image observations and a sparse task completion signal when the cable is correctly inserted. MELD achieves this result by meta-training a latent dynamics model to capture task and state information, as well a policy that conditions on this information to explore and identify the correct insertion point.

inserting a peg into varying targets, and putting away mugs of unknown weights to varying locations on a shelf. We then analyze MELD's capability to meta-learn temporally-extended exploration strategies when only a sparse task completion signal is available. Finally, using a real WidowX robotic arm, we find that after eight hours of meta-training MELD successfully performs Ethernet cable insertion into ports at novel locations and orientations (Figure 19). This real-world experiment with the WidowX is to our knowledge the first demonstration of a meta-RL algorithm trained from images on a real robotic platform. Our open-source implementation of MELD can be found at `https://github.com/tonyzhaozh/meld`.

5.3.1 *Preliminaries: Meta-RL and Latent State Models*

In this work, we leverage tools from latent state modeling to design an efficient meta-RL method that can operate in the real world from image observations. In this section, we review latent state models and meta-RL, developing a formalism that will allow us to derive our algorithm in Section 5.3.2.

Latent State Models See Chapter 3 for preliminaries on POMDPs. To tackle the problem of partial observability, and the difficulty of processing high-dimensional observations such as images, existing approaches (Igl et al., 2018; Gelada et al., 2019; A. Lee et al., 2019) train latent state models to learn meaningful representations of the incoming observations by explicitly representing the unknown Markovian state as a hidden

variable $\mathbf{s}_t$ in a graphical model, as shown in Figure 20 (a). The parameters of these graphical models can be trained by approximately maximizing the log-likelihood of the observations: $\log p(\mathbf{x}_{1:T}|\mathbf{a}_{1:T-1}) = \log \int p(\mathbf{x}_T|\mathbf{s}_T)p(\mathbf{s}_T|\mathbf{s}_{T-1}, \mathbf{a}_{T-1})...p(\mathbf{s}_1)dz$. Given a history of observations and actions seen so far, the posterior distribution over the hidden variable captures the agent's belief over the current underlying state, and can be written as $\mathbf{b}_t = p(\mathbf{s}_t|\mathbf{x}_{1:t}, \mathbf{a}_{1:t-1})$. Then, rather than conditioning the policy on raw observations, these methods learn a policy $\pi(\mathbf{a}_t|\mathbf{b}_t)$ as a function of this belief state.

Meta-Reinforcement Learning See Chapter 3 for the partially observed meta-RL problem statement. We recap the meta-training objective here.

$$\max_{\theta,\phi} \mathop{\mathbb{E}}_{\mathcal{T}\sim p(\mathcal{T})} \mathop{\mathbb{E}}_{\substack{\mathbf{x}_t\sim p_{\mathcal{T}}(\cdot|\mathbf{s}_t)\\ \mathbf{a}_t\sim\pi_\theta(\cdot|\mathbf{x}_t,\mathbf{c}_t)\\ \mathbf{s}_{t+1}\sim p_{\mathcal{T}}(\cdot|\mathbf{s}_t,\mathbf{a}_t)\\ r_t\sim r_{\mathcal{T}}(\cdot|\mathbf{s}_t,\mathbf{a}_t)}} \left[\sum_{t=1}^{T} \gamma^t r_t\right] \quad \text{where} \quad \mathbf{c}_t = f_\phi(\mathbf{x}_{1:t}, r_{1:t}, \mathbf{a}_{1:t-1}). \tag{12}$$

Meta-RL methods may differ in how the adaptation procedure f_ϕ is represented (e.g., as probabilistic inference (Rakelly et al., 2019; Zintgraf et al., 2019), as a recurrent update (Duan et al., 2016; Wang and Hebert, 2016), as a gradient step (Finn et al., 2017a)), how often the adaptation procedure occurs (e.g., at every timestep (Duan et al., 2016; Zintgraf et al., 2019) or once per episode (Rakelly et al., 2019; Humplik et al., 2019)), and also in how the optimization is performed (e.g., on-policy (Duan et al., 2016), off-policy (Rakelly et al., 2019)). Differences aside, these methods all typically optimize this objective end-to-end, creating a representation learning bottleneck when learning from image inputs that are ubiquitous in real-world robotics. In the following section, we show how the latent state models discussed in this section can be re-purposed for joint representation and task learning, and how this insight leads to a practical algorithm for image-based meta-RL.

5.3.2 *MELD Algorithm*

In this section, we present MELD: an efficient algorithm for meta-RL from images. We first develop the algorithm and then describe its implementation.

Meta-RL with Latent Dynamics Models To see how task inference in meta-RL can be cast as latent state inference, consider the graphical models depicted in Figure 20. Panel (a) illustrates a standard POMDP with underlying latent state $\mathbf{z}_t$ and observations $\mathbf{x}_t$, and panel (b) depicts standard meta-RL, where the hidden task variable $\mathcal{T}$ is assumed constant throughout the episode. In the meta-RL setting, the policy must then be condi-

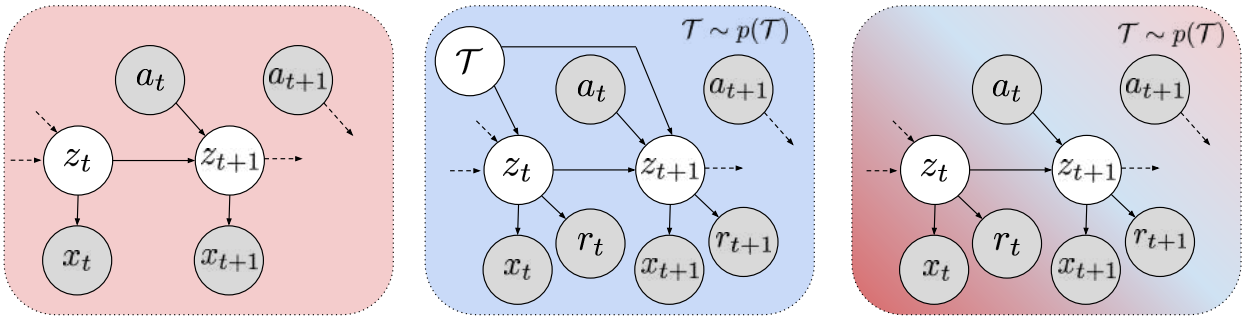


Figure 20: **(a)**: When only partial observations of the underlying state are available, latent dynamics models can glean state information $\mathbf{z}_t$ from a history of observations. **(b)**: Meta-RL considers a task distribution where the current task $\mathcal{T}$ is an unobserved variable that controls dynamics and rewards. **(c)**: We interpret $\mathcal{T}$ as part of $\mathbf{z}_t$, allowing us to leverage latent dynamics models for efficient image-based meta-RL.

tioned on both the observation and the task variable in order to adapt to a new task (see Equation 12). Casting this task variable as part of the latent state, panel (c) illustrates our graphical model, where the states $\mathbf{s}_t$ now contain both state and task information. In effect, we cast the task distribution over POMDPs as a POMDP itself, where the state variables now additionally capture task information. This meld allows us to draw on the rich literature of latent state models discussed in Section 5.3.1, and use them here to tackle the problem of meta-RL from sensory observations. Note that the task is not explicitly handled since it is simply another hidden state variable, providing a seamless integration of meta-RL with learning from sensory observations.

Concretely, we learn a latent state model over hidden variables by optimizing the log-likelihood of the evidence (observations and rewards) in the graphical model in Figure 20c:

$$\max_{\phi} \mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})} \mathbb{E}_{\substack{\mathbf{x}_t\sim p_{\mathcal{T}}(\cdot|\mathbf{s}_t)\,,\,\mathbf{a}_t\sim\pi_\theta(\cdot|\mathbf{b}_t) \\ \mathbf{s}_{t+1}\sim p_{\mathcal{T}}(\cdot|\mathbf{s}_t,\mathbf{a}_t)\,,\,r_t\sim r_{\mathcal{T}}(\cdot|\mathbf{s}_t,\mathbf{a}_t)}} \left[\log p_\phi(\mathbf{x}_{1:T}, r_{1:T}|\mathbf{a}_{1:T-1})\right]. \tag{13}$$

Note that the only change from the latent state model from Section 5.3.1 is the inclusion of rewards as part of the observed evidence. While this change appears simple, it enables meta-learning by allowing the hidden state to capture task information. Posterior inference in this model then gives the agent's belief $\mathbf{b}_t = p(\mathbf{s}_t|\mathbf{x}_{1:t}, r_{1:t}, \mathbf{a}_{1:t-1})$ over latent state and task variables $\mathbf{s}_t$. Conditioned on this belief, the policy $\pi_\theta(\mathbf{a}_t|\mathbf{b}_t)$ can learn to adapt its behavior to the task. Prescribing the adaptation procedure f_ϕ from Equation 12 to be posterior inference in our latent state model, the meta-training objective in MELD

is:

$$\max_{\theta} \mathop{\mathbb{E}}_{\mathcal{T}\sim p(\mathcal{T})} \mathop{\mathbb{E}}_{\substack{\mathbf{x}_t\sim p_{\mathcal{T}}(\cdot|\mathbf{s}_t)\,,\,\mathbf{a}_t\sim\pi_\theta(\cdot|\mathbf{b}_t)\\ \mathbf{s}_{t+1}\sim p_{\mathcal{T}}(\cdot|\mathbf{s}_t,\mathbf{a}_t)\,,\,r_t\sim r_{\mathcal{T}}(\cdot|\mathbf{s}_t,\mathbf{a}_t)}} \left[\sum_{t=1}^{T}\gamma^t r_t\right] \tag{14}$$
$$\text{where } \mathbf{b}_t = p(\mathbf{s}_t|\mathbf{x}_{1:t}, r_{1:t}, \mathbf{a}_{1:t-1})$$

By melding state and task inference, MELD inherits the same representation learning mechanism as latent state models discussed in Section 5.3.1 to enable efficient meta-RL with images.

Implementing MELD Exactly computing the posterior distribution is intractable, so we take a variational inference approach (Wainwright and Jordan, 2008) to maximize a lower bound on the objective in Equation 13. We factorize the variational posterior as $q(\mathbf{s}_{1:T}|\mathbf{x}_{1:T}, r_{1:T}, \mathbf{a}_{1:T-1}) = q(\mathbf{s}_T|\mathbf{x}_T, r_T, \mathbf{s}_{T-1}, \mathbf{a}_{T-1}) \dots q(\mathbf{s}_2|\mathbf{x}_2, r_2, \mathbf{s}_1, \mathbf{a}_1) q(\mathbf{s}_1|\mathbf{x}_1, r_1)$. With this factorization, we implement each component as a deep neural network and optimize the evidence lower bound of the joint objective, where $\mathbb{E}_{\mathbf{s}_{1:t}\sim q_\phi}[\log p(\mathbf{x}_{1:T}, r_{1:T}|\mathbf{a}_{1:T-1})] \geqslant \mathcal{L}_{\text{model}}$, with $\mathcal{L}_{\text{model}}$ defined as:

$$\mathcal{L}_{\text{model}}(\mathbf{x}_{1:T}, r_{1:T}, \mathbf{a}_{1:T-1}) = \mathop{\mathbb{E}}_{\mathbf{s}_{1:T}\sim q_\phi} \sum_{t=1}^{T} \log p_\phi(\mathbf{x}_t|\mathbf{s}_t) + \log p_\phi(r_t|\mathbf{s}_t)$$
$$- D_{KL}(q_\phi(\mathbf{s}_1|\mathbf{x}_1, r_1)\|p(\mathbf{s}_1)) - \sum_{t=2}^{T} D_{KL}(q_\phi(\mathbf{s}_t|\mathbf{x}_t, r_t, \mathbf{s}_{t-1}, \mathbf{a}_{t-1})\|p_\phi(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1})). \tag{15}$$

The first two terms encourage a rich latent representation $\mathbf{s}_t$ by requiring reconstruction, while the last term keeps the inference network consistent with latent dynamics. The first timestep posterior $q_\phi(\mathbf{s}_1|\mathbf{x}_1, r_1)$ is modeled separately from the remaining steps, and $p(\mathbf{s}_1)$ is chosen to be a fixed unit Gaussian $\mathcal{N}(0, I)$. The learned inference networks $q_\phi(\mathbf{s}_1|\mathbf{x}_1, r_1)$ and $q_\phi(\mathbf{s}_t|\mathbf{x}_t, r_t, \mathbf{s}_{t-1}, \mathbf{a}_{t-1})$, decoder networks $p_\phi(\mathbf{x}_t|\mathbf{s}_t)$ and $p_\phi(r_t|\mathbf{s}_t)$, and dynamics $p_\phi(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1})$ are all fully connected networks that output parameters of Gaussian distributions. We follow the architecture of the latent variable model from SLAC (A. Lee et al., 2019) and provide the remaining implementation details in Appendix D.1.

We use the soft actor-critic (SAC) (Haarnoja et al., 2018) RL algorithm in this work, due to its high sample efficiency and performance. The actor $\pi_\theta(\mathbf{a}_t|\mathbf{b}_t)$ and the critic $Q_\psi(\mathbf{b}_t, \mathbf{a}_t)$ are conditioned on the posterior belief $\mathbf{b}_t$, modeled as fully connected neural networks, and trained as prescribed by the SAC algorithm. During meta-training, MELD alternates between collecting data with the current policy, training the model by

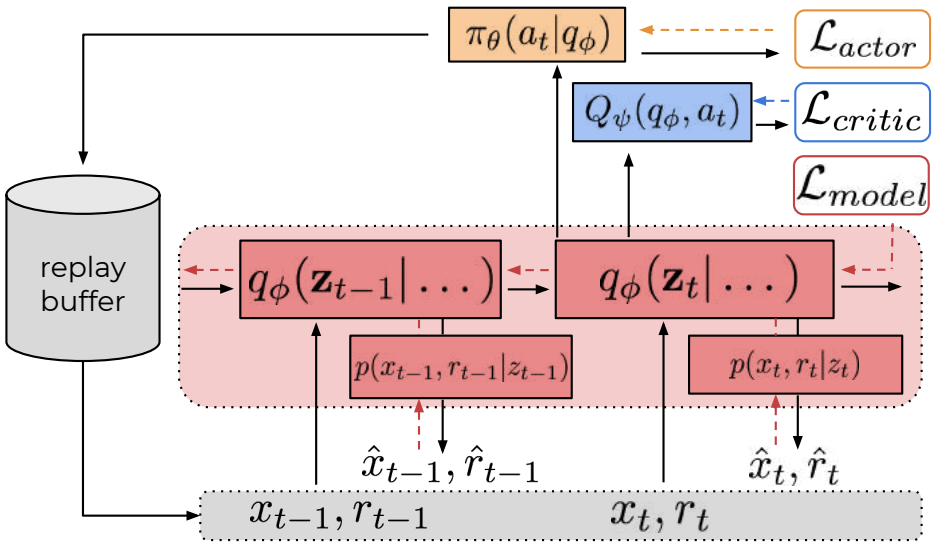


Figure 21: MELD meta-training alternates between collecting data with π_θ and training the latent state model p_ϕ, inference networks q_ϕ, actor π_θ, and critic Q_ζ.

optimizing $\mathcal{L}_{\text{model}}$, and training the policy with the current model. Meta-training and meta-testing are described in Algorithm 3 and 4 respectively.

Algorithm 3 MELD Meta-training

Require: Training tasks $\{\mathcal{T}_i\}_{i=1\ldots J}$ from $p(\mathcal{T})$, learning rates η_1, η_2, η_3

Initialize model p_ϕ, q_ϕ, actor π_θ, critic Q_ζ
Initialize replay buffers $\mathcal{B}_i$ for each training task
while not done **do**
  for each task $\mathcal{T}_i$ **do** ▷ collect data
    Infer belief $\mathbf{b}_1 = q_\phi(\mathbf{s}_1|\mathbf{x}_1, r_1)$
    Step environment with $\mathbf{a}_1 \sim \pi_\theta(\mathbf{a}|\mathbf{b}_1)$, get $\mathbf{x}_2$, r_2
    for $t = 2, \ldots T-1$ **do**
      Infer belief over latent state $\mathbf{b}_t = q_\phi(\mathbf{z}_t|\mathbf{x}_t, r_t, \mathbf{z}_{t-1}, \mathbf{a}_{t-1})$
      Step environment with $\mathbf{a}_t \sim \pi_\theta(\mathbf{a}|\mathbf{b}_t)$, get $\mathbf{x}_{t+1}$, r_{t+1}
    end for
    Add data $\{\mathbf{x}_{1:T}, r_{1:T}, \mathbf{a}_{1:T-1}\}$ to replay buffer $\mathcal{B}^i$
  end for
  for step in train steps **do**
    for each task $\mathcal{T}_i$ **do**
      Sample transitions from task-buffer $\{\mathbf{x}_{1:T}, r_{1:T}, \mathbf{a}_{1:T-1}\} \sim \mathcal{B}_i$
      Infer beliefs at each time step $\mathbf{b}_{1:T} = \{q_\phi(\mathbf{s}_t|\ldots)\}_{1:T}$
      Predict observation and reward reconstructions $\{\hat{\mathbf{x}}_t, \hat{r}_t\}_{1:T}$
      Compute model loss$\mathcal{L}_m^i = \mathcal{L}_{model}(\{\mathbf{x}_t, \hat{\mathbf{x}}_t, r_t, \hat{r}_t, \mathbf{a}_t\}_{1:T})$
    end for
    $\phi \leftarrow \phi - \eta_1 \nabla_\phi \sum_i \mathcal{L}_m^i$ ▷ train model
    Update θ, ζ with SAC(η_2, η_3) ▷ train AC
  end for
end while

Algorithm 4 MELD Meta-testing

Require: Test task $\mathcal{T} \sim p(\mathcal{T})$

Infer belief over latent state $\mathbf{b}_1 = q_\phi(\mathbf{z}_1|\mathbf{x}_1, r_1)$
Step environment with $\mathbf{a}_1 \sim \pi_\theta(\mathbf{a}|\mathbf{b}_1)$, get $\mathbf{x}_2$, r_2
for $t = 2, \ldots, T-1$ **do**
  Infer belief $\mathbf{b}_t = q_\phi(\mathbf{z}_t|\mathbf{x}_t, r_t, \mathbf{z}_{t-1}, \mathbf{a}_{t-1})$
  Step environment with $\mathbf{a}_t \sim \pi_\theta(\mathbf{a}|\mathbf{b}_t)$, get $\mathbf{x}_{t+1}$, r_{t+1}
end for

5.3.3 *Experiments*

In our experiments, we aim to answer the following: **(1)** How does MELD compare to prior meta-RL methods in enabling fast acquisition of new skills at test time in challenging simulated control problems? **(2)** Can MELD meta-learn effective exploration when only sparse task completion rewards are available at meta-test time? **(3)** Can MELD enable real robots to quickly acquire skills via meta-RL from images?

MELD in Simulated Environments In this section, we evaluate MELD on the four simulated image-based continuous control problems in Figure 22. In (a) Cheetah-vel, each task is a different target running velocity for the 6-DoF legged robot. Reward is the difference in robot velocity from the target. The remainder of the problems use a 7-DOF Sawyer robotic arm. In (b) Reacher, each task is a different goal position for the end-effector. In (c) Peg-insertion, the robot must insert the peg into the correct box, where each task varies the goal box as well as locations of all four boxes. In (d) Shelf-placing, each task varies the weight (dynamics change) and target location (reward change) of a mug that the robot must move to the shelf. For the Sawyer environments, the reward function is the negative distance between the robot end-effector and the desired end location. For all environments, we train with 30 meta-training tasks and evaluate on 10 meta-test tasks from the same distribution that are not seen during training.

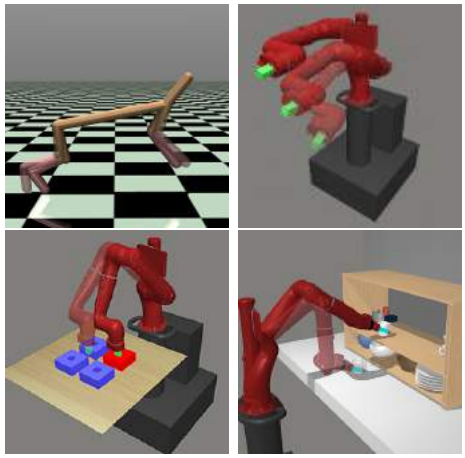


Figure 22: Simulated locomotion and manipulation meta-RL environments in the MuJoCo simulator (Todorov et al., 2012); goals illustrated for visualization purposes.

We compare MELD to two representative state-of-the-art meta-RL algorithms, the algorithm PEARL (Rakelly et al., 2019) proposed in Section 5.2 and RL^2 (Duan et al., 2016). PEARL models a belief over a probabilistic latent task variable as a function of un-ordered batches of transitions, and conditions the policy on both the current observation and this inferred task belief. Unlike MELD, this algorithm assumes an exploration phase of several trajectories in the new task to gather information before adapting, so to get its best performance, we evaluate only *after* this exploration phase. RL^2 models the policy as a recurrent network that directly maps observations, actions, and rewards to actions. To apply PEARL and RL^2 with image observations, we augment them with the

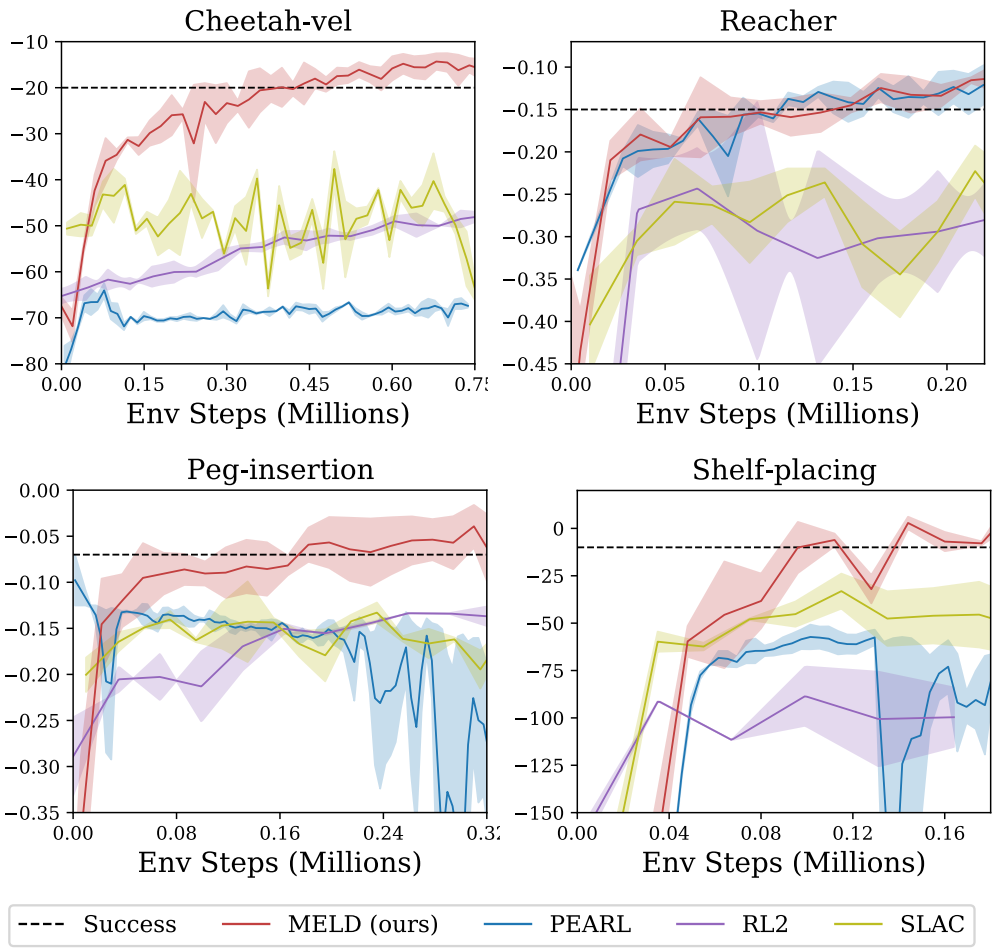


Figure 23: Rewards on test tasks versus meta-training environment steps, comparing MELD to prior methods. Black dashed line indicates consistent task success (see Appendix D.2 for definition).

same convolutional encoder architecture used by MELD. Finally, to verify the need for task inference to solve new tasks, we compare to SLAC (A. Lee et al., 2019), which infers state information from a sequence of observations but does *not* perform meta-learning.

In Figure 23 we plot average performance on meta-test tasks over the course of meta-training, across 3 random seeds. See Appendix D.2 for the the definitions of metrics used for each task. MELD achieves the highest performance in each environment and is the only method to fully solve Cheetah-vel, Peg-insertion, and Shelf-placing. The SLAC baseline fails in this meta-RL setting, as expected, with qualitative behavior of always executing a single "average" motion, such as reaching toward a mean goal location and running at a medium speed. PEARL aggregates task information over time in its latent task variable, but relies on the current observation alone for state information. Its poor

performance on Cheetah, Reacher, and Shelf reflect the need for state estimation from a sequence of observations to perform control in these environments. While RL^2 is capable of propagating both state and task information over time, we observe that it overfits heavily to training tasks and struggles on evaluation tasks.

Temporally-Extended Exploration The previous section assumed a shaped reward function that is the negative distance between the current robot position and the desired one at every timestep. In the real world, this type of reward function is typically not available to the agent, since it requires information that may be difficult or impossible to obtain. For example, in the Ethernet cable insertion problem, the location of the insertion point is unknown, but the agent might receive a sparse task completion reward upon making the correct electrical connection. To quickly succeed at a new task given only this sparse signal at meta-test time, it is critical for MELD to reason over multiple episodes and acquire temporally-extended strategies during meta-training. Because RL with sparse rewards is very inefficient, we follow prior work (Gupta et al., 2018b; Rakelly et al., 2019) and assume access to a shaped reward function during meta-training to help learn these strategies. We detail our particular approach to making use of shaped rewards during meta-training in Appendix D.3, and at meta-test time assume access to only the sparse reward signal.

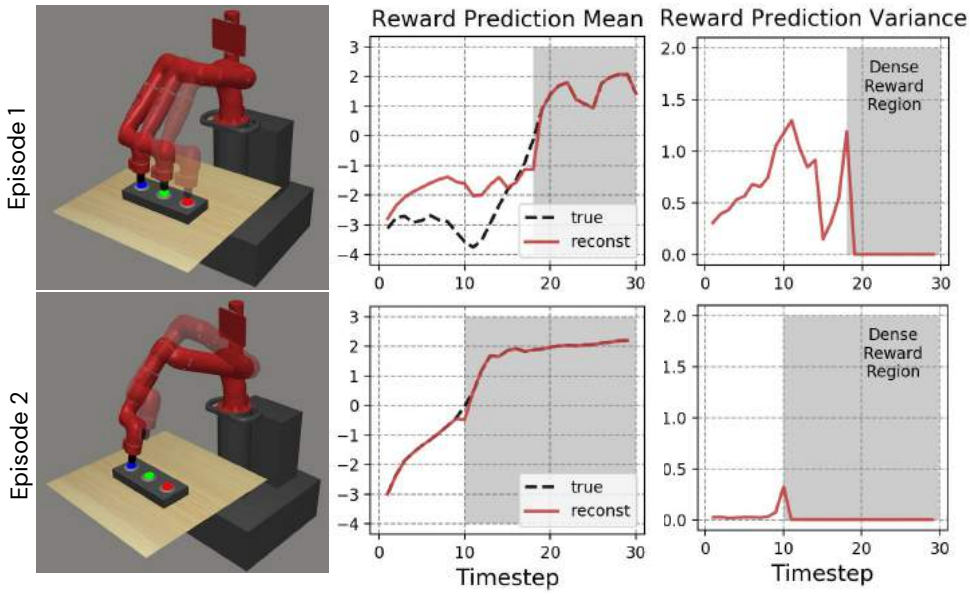


Figure 24: Button press with reward given only upon pressing correct button. The robot explores each button until it finds the correct one (top) and returns to that button immediately in the next episode (bottom). See text for discussion.

To evaluate MELD in this setting, we design a simulated button-pressing environment with the Sawyer robot, where the button to push and the location of the panel changes with each task. Sparse reward is given only when the correct button is pushed, while shaped reward (used only in meta-training) is the negative distance from the robot's end-effector to the insertion point. In Figure 24, we analyze the qualitative behavior of MELD when learning a new task at test time. Though the shaped reward is not used at test time, we plot the shaped reward reconstruction mean and variance to gain an insight into the contents of the learned latent state. In the first episode, predicted reward error and variance are high until the robot presses the correct button. MELD's latent state model persists the task information to the second episode (predicted reward error and variance are very low) and the robot navigates immediately to the correct button. In Figure 25, we compare MELD to the same baselines introduced in the previous section and find that it is the only method able to press the correct button within two trajectories of experience. In the next section, we test MELD's capability to perform such exploration and exploitation in the real world.

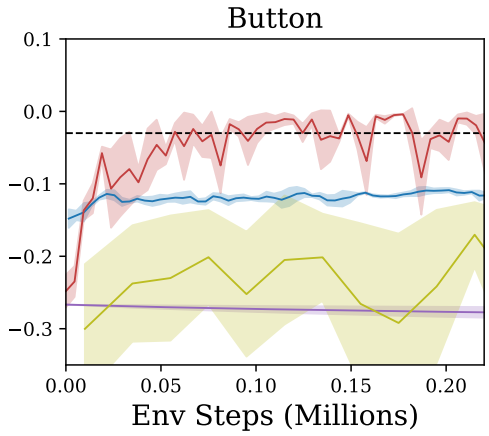


Figure 25: Comparison to approaches from Figure 23 on the button-pressing environment, showing test task performance versus meta-training environment steps.

MELD in the Real World We now evaluate MELD on a real-world 5-DoF WidowX arm performing Ethernet cable insertion. The task distribution consists of different ports in a router that also varies in location and orientation (see Figure 26). To instrument these tasks in the real world, we build an automatic reset mechanism that moves and rotates the router, as detailed in Appendix D.5. At meta-test time the reward is a sparse signal given when the robot inserts the cable in the correct port. As in the previous section, during meta-training we make use of a shaped reward function that is the sum of the L2-norms of translational and rotational distances between the pose of the object in the end-effector and a goal pose. The agent's observations are concatenated images from two webcams (Figure 27): one fixed view and one first-person view from a wrist-mounted camera. The policy sends joint velocity controls over a ROS interface to a low-level PID controller to move the joints of each robot.

We compare MELD to SLAC as described in Section 5.3.3 as well as a random policy, and plot the results in Figure 28. After training across 20 meta-training tasks using a total of 8 hours (80,000 samples at 3.3Hz) worth of data, MELD achieves a success rate of 90% over three rounds of evaluation in each of the 10 randomly sampled evaluation tasks that were not seen during training. To our knowledge, this experiment is the first demonstration of meta-RL trained entirely in the real world from image observations. We also conducted experiments with the Sawyer robot, finding that MELD enables the Sawyer to insert a peg into the correct hole given a per-timestep reward of distance to the hole (see Appendix D.6). Due to lab access restrictions as a result of COVID-19, we could not evaluate adaptation to new tasks on this platform. Videos of all experiments can be found on our project website.[1]

Figure 26: Ethernet cable insertion problem. The automatic reset mechanism changes the position and orientation of the router across tasks.

Figure 27: 64x128 image observations seen by WidowX.

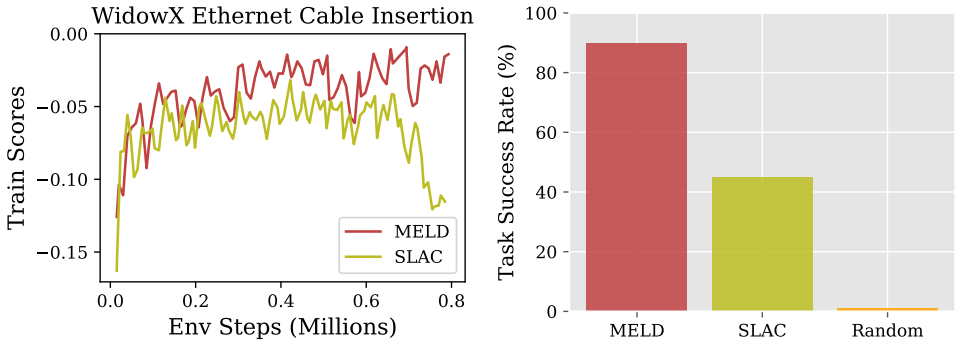


Figure 28: **(left)** Training task rewards versus meta-training environment steps. **(right)** Task success (defined as full insertion into correct hole) on held-out tasks after meta-training.

1 https://sites.google.com/view/meld-lsm/home

5.3.4 *Discussion*

In this chapter, we leveraged the insight that meta-RL can be cast into the framework of latent state inference. This allows us to combine the fast skill-acquisition capabilities of meta-learning with the efficiency of unsupervised latent state models when learning from raw image observations. Based on this principle, we designed MELD, a practical algorithm for meta-RL with image observations. MELD outperforms prior methods on simulated locomotion and manipulation tasks, and is efficient enough to perform meta-RL directly from images in the real world.

However, neither our approach nor prior meta-learning works have shown convincing generalization to wider task distributions of qualitatively distinct manipulation tasks. A significant stumbling block is the difficulty of designing reward functions and instrumenting distinct tasks in the real world. Instead of relying solely on human-designed reward functions for learning, agents can also make use of unsupervised skill learning (Gregor et al., 2016; Eysenbach et al., 2018) to acquire knowledge about their environment, and even use these skills as tasks for meta-learning algorithms (Gupta et al., 2018a). These representations can encode coherent temporally-extended behaviors, such as reaching for objects, reducing the burden on reward function design to encourage these behaviors. The need to reset the environment at the start of every episode in RL also poses major challenges, particularly when there are many distinct tasks to reset. While there has been some work studying reset-free RL (Eysenbach et al., 2017; Zhu et al., 2020; Lu et al., 2020), the solution is far from clear and we see this as a promising direction for future development. Without the need for manual resets and task instrumentation, meta-RL algorithms could be applied to a much wider variety of real-world robotic problems, enabling agents to acquire qualitatively new skills from little experience.

6

CONCLUSION

In this thesis, we considered the problem of how artificial agents can learn faster and with less supervision. We characterized the problem as one of learning appropriate abstractions from data. We began with learning abstractions of states via unsupervised representation learning (Chapter 2), and proved that several popular state representation learning objectives are not guaranteed to yield representations sufficient for optimal control, while a third is sufficient. This criteria of sufficiency is equally relevant to representations of tasks. As unsupervised skill-learning algorithms are proposed to reduce the burden of reward supervision, they should be evaluated in terms of their ability to preserve information needed to solve new tasks.

In Chapter 3, we turned to the problem of learning representations of tasks. Given a mechanism to infer task representations, and a meta-learned policy that acts according to the task representation, the agent can acquire new skills rapidly via inference. Working first in the supervised learning setting and with deterministic task variables, in Chapter 4 we proposed an algorithm for meta-learned image segmentation of new concepts. Moving to the reinforcement learning setting, in Chapter 5 we extended this perspective to probabilistic task variables to enable the agent to reason about uncertainty. We designed an efficient off-policy algorithm PEARL, and an algorithm MELD for meta-RL from image observations. These works demonstrate that viewing meta-RL as inference of probabilistic task abstractions can enable simple and practical solutions to challenges such as off-policy learning, efficient exploration, and learning from images.

We believe that future work in learning data-driven abstractions might focus on three key areas: (1) understanding theoretically the properties guaranteed and encouraged by abstraction-learning objectives, (2) developing more effective methods for multi-task optimization, and (3) handling varying amounts and type of supervision in new tasks. For the first, while sufficiency is a basic and necessary criterion, there are many other criteria desirable in an abstraction, such as minimality, noise-robustness, interpretability,

composability, etc. Theoretical guarantees regarding such desiderata may greatly help in guiding research in new objectives for abstraction learning. For the second, recent evidence seems to suggest that multi-task optimization is a major stumbling block for performing meta-RL with broader task distributions (Schaul et al., 2019; Yu et al., 2019a; Yu et al., 2020). In our own experiment in Figure 14, while PEARL achieves a state-of-the-art result in terms of return on the humanoid domain, we found that qualitatively the robot fails to actually learn to locomote at all, even when the tasks differ only in which direction to walk. Perhaps a curriculum approach, or advancements in optimization for off-policy deep RL, would alleviate these optimization issues, enabling meta-RL agents to meta-learn qualitatively different behaviors. For the third, meta-learning algorithms generally work for the amount and type of supervision provided during meta-training. However, realistic scenarios are not divided cleanly into few-shot (meta-learning) and many-shot (learning) paradigms. Instead, there exists a spectrum of amounts and types of supervision that agents must be able to learn from throughout their lifetimes. A unifying framework for few to many-shot learning would provide a principled way for agents to learn and consolidate knowledge.

BIBLIOGRAPHY

Abel, David, D Ellis Hershkowitz, and Michael L Littman (2016). "Near optimal behavior via approximate state abstraction." In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*, pp. 2915–2923 (cit. on p. 6).

Agrawal, Pulkit, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine (2016). "Learning to poke by poking: Experiential learning of intuitive physics." In: *Advances in neural information processing systems*, pp. 5074–5082 (cit. on pp. 6, 10).

Alemi, Alexander A, Ian Fischer, Joshua V Dillon, and Kevin Murphy (2016). "Deep variational information bottleneck." In: *arXiv preprint arXiv:1612.00410* (cit. on p. 42).

Anand, Ankesh, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm (2019). "Unsupervised state representation learning in atari." In: *Advances in Neural Information Processing Systems*, pp. 8766–8779 (cit. on pp. 4, 6, 10).

Arndt, Karol, Murtaza Hazara, Ali Ghadirzadeh, and Ville Kyrki (2019). "Meta Reinforcement Learning for Sim-to-real Domain Adaptation." In: *arXiv preprint arXiv:1909.12906* (cit. on p. 40).

Bachman, Philip, R Devon Hjelm, and William Buchwalter (2019). "Learning representations by maximizing mutual information across views." In: *Advances in Neural Information Processing Systems*, pp. 15509–15519 (cit. on p. 5).

Bean, James C, John R Birge, and Robert L Smith (1987). "Aggregation in dynamic programming." In: *Operations Research* 35.2, pp. 215–220 (cit. on pp. 6, 8).

Bearman, Amy, Olga Russakovsky, Vittorio Ferrari, and Li Fei-Fei (2016). "What's the point: Semantic segmentation with point supervision." In: *ECCV* (cit. on p. 25).

Becker, Suzanna and Geoffrey E Hinton (1992). "Self-organizing neural network that discovers surfaces in random-dot stereograms." In: *Nature* 355.6356, pp. 161–163 (cit. on p. 5).

Belghazi, Mohamed Ishmael, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R Devon Hjelm (2018). "Mine: mutual information neural estimation." In: *arXiv preprint arXiv:1801.04062* (cit. on pp. 4, 5).

Bell, Anthony J and Terrence J Sejnowski (1995). "An information-maximization approach to blind separation and blind deconvolution." In: *Neural computation* 7.6, pp. 1129–1159 (cit. on pp. 4, 5).

Bellemare, Marc, Will Dabney, Robert Dadashi, Adrien Ali Taiga, Pablo Samuel Castro, Nicolas Le Roux, Dale Schuurmans, Tor Lattimore, and Clare Lyle (2019). "A geometric perspective on optimal representations for reinforcement learning." In: *Advances in Neural Information Processing Systems*, pp. 4360–4371 (cit. on p. 6).

Bengio, Yoshua, Samy Bengio, and Jocelyn Cloutier (1990). *Learning a synaptic learning rule*. Université de Montréal (cit. on pp. 2, 37).

Bertinetto, Luca, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi (2016). "Learning feed-forward one-shot learners." In: *NIPS* (cit. on p. 24).

Bertsekas, Dimitri P, David A Castanon, et al. (1988). "Adaptive aggregation methods for infinite horizon dynamic programming." In: (cit. on p. 6).

Bicchi, Antonio and Vijay Kumar (2000). "Robotic grasping and contact: A review." In: *ICRA*. Vol. 1. IEEE, pp. 348–353 (cit. on p. 40).

Bonardi, Alessandro, Stephen James, and Andrew J Davison (2019). "Learning One-Shot Imitation from Humans without Humans." In: *arXiv preprint arXiv:1911.01103* (cit. on p. 40).

Boykov, Yuri Y and M-P Jolly (2001). "Interactive graph cuts for optimal boundary & region segmentation of objects in ND images." In: *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*. Vol. 1. IEEE, pp. 105–112 (cit. on p. 24).

Caelles, Sergi, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool (2017). "One-Shot Video Object Segmentation." In: *CVPR* (cit. on pp. 24, 28, 31).

Chen, Yuhua, Jordi Pont-Tuset, Alberto Montes, and Luc Van Gool (2018). "Blazingly Fast Video Object Segmentation with Pixel-Wise Metric Learning." In: *CVPR*, pp. 1189–1198 (cit. on pp. 24, 32).

Chopra, Sumit, Raia Hadsell, and Yann LeCun (2005). "Learning a similarity metric discriminatively, with application to face verification." In: *CVPR*. Vol. 1. IEEE, pp. 539–546 (cit. on p. 24).

Cover, Thomas M (1999). *Elements of information theory*. John Wiley & Sons (cit. on p. 7).

Daumé III, Hal (2009). "Bayesian multitask learning with latent hierarchies." In: *arXiv preprint arXiv:0907.0783* (cit. on p. 3).

Dean, Thomas and Robert Givan (1997). "Model minimization in Markov decision processes." In: *AAAI/IAAI*, pp. 106–111 (cit. on p. 6).

Deisenroth, Marc Peter and Jan Peters (2012). "Solving nonlinear continuous state-action-observation POMDPs for mechanical systems with Gaussian noise." In: (cit. on p. 40).

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186 (cit. on p. 5).

Donahue, Jeffrey, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell (2015). "Long-term recurrent convolutional networks for visual recognition and description." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2625–2634 (cit. on p. 1).

Doshi-Velez, Finale and George Konidaris (2016). "Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations." In: *IJCAI: proceedings of the conference*. Vol. 2016. NIH Public Access, p. 1432 (cit. on p. 38).

Dosovitskiy, Alexey and Vladlen Koltun (2016). "Learning to Act by Predicting the Future." In: *International Conference on Learning Representations* (cit. on p. 6).

Du, Simon S, Sham M Kakade, Ruosong Wang, and Lin F Yang (2019). "Is a Good Representation Sufficient for Sample Efficient Reinforcement Learning?" In: *arXiv preprint arXiv:1910.03016* (cit. on pp. 6, 9).

Duan, Yan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba (2017). "One-shot imitation learning." In: *NeurIPS*, pp. 1087–1098 (cit. on p. 38).

Duan, Yan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel (2016). "RL2: Fast Reinforcement Learning via Slow Reinforcement Learning." In: *arXiv preprint arXiv:1611.02779* (cit. on pp. 2, 20, 36, 38, 45, 48, 51, 55, 60).

Edwards, Harrison and Amos Storkey (2016). "Towards a neural statistician." In: *arXiv preprint arXiv:1606.02185* (cit. on p. 3).

Everingham, Mark, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman (2010). "The pascal visual object classes (voc) challenge." In: *International journal of computer vision* 88.2, pp. 303–338 (cit. on pp. 24, 31, 33, 34, 91).

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 p. 65).

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 p. 65).

Fedus, William, Carles Gelada, Yoshua Bengio, Marc G Bellemare, and Hugo Larochelle (2019). "Hyperbolic discounting and learning over multiple horizons." In: *arXiv preprint arXiv:1902.06865* (cit. on p. 6).

Fei-Fei, Li et al. (2003). "A Bayesian approach to unsupervised one-shot learning of object categories." In: *Proceedings Ninth IEEE International Conference on Computer Vision*. IEEE, pp. 1134–1141 (cit. on p. 39).

Fei-Fei, Li, Rob Fergus, and Pietro Perona (2006). "One-shot learning of object categories." In: *PAMI* (cit. on pp. 3, 23, 25).

Ferns, Norm, Pablo Samuel Castro, Doina Precup, and Prakash Panangaden (2006). "Methods for computing state similarity in Markov decision processes." In: *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, pp. 174–181 (cit. on p. 6).

Finn, Chelsea, Pieter Abbeel, and Sergey Levine (2017a). "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks." In: *ICML* (cit. on pp. 2, 17, 18, 20, 24, 35, 36, 38, 45, 48, 55).

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 p. 2).

Finn, Chelsea, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel (2016). "Deep spatial autoencoders for visuomotor learning." In: *ICRA*. IEEE, pp. 512–519 (cit. on pp. 37, 40).

Finn, Chelsea, Kelvin Xu, and Sergey Levine (2018). "Probabilistic model-agnostic meta-learning." In: *Advances in Neural Information Processing Systems*, pp. 9537–9548 (cit. on pp. 36, 39).

Finn, Chelsea, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine (2017b). "One-Shot Visual Imitation Learning via Meta-Learning." In: *CoRL*, pp. 357–368 (cit. on p. 40).

Florence, Peter, Lucas Manuelli, and Russ Tedrake (2019). "Self-supervised correspondence in visuomotor policy learning." In: *IEEE Robotics and Automation Letters* 5.2, pp. 492–499 (cit. on p. 40).

Garcia-Garcia, Alberto, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and Jose Garcia-Rodriguez (2017). "A review on deep learning techniques applied to semantic segmentation." In: *arXiv preprint arXiv:1704.06857* (cit. on p. 24).

Garcia, Victor and Joan Bruna (2018). "Few-Shot Learning with Graph Neural Networks." In: *ICLR* (cit. on p. 25).

Gelada, Carles, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare (2019). "DeepMDP: Learning Continuous Latent Space Models for Representation Learning." In: *ICML*, pp. 2170–2179 (cit. on pp. 6, 40, 54).

Ghadirzadeh, Ali, Atsuto Maki, Danica Kragic, and Mårten Björkman (2017). "Deep predictive policy training using reinforcement learning." In: *IROS*. IEEE, pp. 2351–2358 (cit. on pp. 37, 40).

Givan, Robert, Thomas Dean, and Matthew Greig (2003). "Equivalence notions and model minimization in Markov decision processes." In: *Artificial Intelligence* 147.1-2, pp. 163–223 (cit. on p. 6).

Gordon, Jonathan, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard Turner (2019). "Meta-Learning Probabilistic Inference for Prediction." In: *International Conference on Learning Representations* (cit. on pp. 36, 39).

Grant, Erin, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths (2018). "Recasting gradient-based meta-learning as hierarchical bayes." In: *International Conference on Learning Representations* (cit. on p. 39).

Gregor, Karol and Frederic Besse (2018). "Temporal difference variational auto-encoder." In: *arXiv preprint arXiv:1806.03107* (cit. on p. 40).

Gregor, Karol, Danilo Jimenez Rezende, and Daan Wierstra (2016). "Variational intrinsic control." In: *arXiv preprint arXiv:1611.07507* (cit. on pp. 6, 10, 65).

Gullapalli, Vijaykumar, Judy A Franklin, and Hamid Benbrahim (1994). "Acquiring robot skills via reinforcement learning." In: *IEEE CSM* 14.1, pp. 13–24 (cit. on p. 40).

Gupta, Abhishek, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine (2018a). "Unsupervised meta-learning for reinforcement learning." In: *arXiv preprint arXiv:1806.04640* (cit. on p. 65).

Gupta, Abhishek, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine (2018b). "Meta-Reinforcement Learning of Structured Exploration Strategies." In: *Advances in Neural Information Processing Systems* (cit. on pp. 39, 41, 45, 50, 62, 100, 102).

Gutmann, Michael and Aapo Hyvärinen (2010). "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models." In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 297–304 (cit. on pp. 5, 14).

Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel, and Sergey Levine (2018). "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor." In: *International Conference on Machine Learning* (cit. on pp. 14, 45, 57, 88, 95, 96).

Hadsell, Raia, Sumit Chopra, and Yann LeCun (2006). "Dimensionality reduction by learning an invariant mapping." In: *CVPR*. Vol. 2. IEEE, pp. 1735–1742 (cit. on p. 24).

Hafner, Danijar, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson (2019). "Learning latent dynamics for planning from pixels." In: *ICML*, pp. 2555–2565 (cit. on pp. 6, 8, 40).

Hariharan, Bharath, Pablo Arbelaez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik (2011). "Semantic Contours from Inverse Detectors." In: *ICCV* (cit. on p. 91).

Hassan, Hany, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. (2018). "Achieving human parity on automatic chinese to english news translation." In: *arXiv preprint arXiv:1803.05567* (cit. on p. 1).

Hausknecht, Matthew and Peter Stone (2015). "Deep Recurrent Q-Learning for Partially Observable MDPs." In: *2015 AAAI Fall Symposium Series* (cit. on pp. 38, 40).

Hausman, Karol, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller (2018). "Learning an embedding space for transferable robot skills." In: *International Conference on Learning Representations* (cit. on p. 39).

Heess, Nicolas, Jonathan J Hunt, Timothy P Lillicrap, and David Silver (2015). "Memory-based control with recurrent neural networks." In: *arXiv preprint arXiv:1512.04455* (cit. on pp. 38, 40, 48, 51).

Henrich, Dominik and Heinz Wörn (2012). *Robot manipulation of deformable objects*. Springer Science & Business Media (cit. on p. 40).

Hjelm, R Devon, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Adam Trischler, and Yoshua Bengio (2018). "Learning deep representations by mutual information estimation and maximization." In: *arXiv preprint arXiv:1808.06670* (cit. on pp. 4, 5).

Houthooft, Rein, Richard Y Chen, Phillip Isola, Bradly C Stadie, Filip Wolski, Jonathan Ho, and Pieter Abbeel (2018). "Evolved policy gradients." In: *Neural Information Processing Systems (NIPS)* (cit. on p. 38).

Humplik, Jan, Alexandre Galashov, Leonard Hasenclever, Pedro A Ortega, Yee Whye Teh, and Nicolas Heess (2019). "Meta reinforcement learning as task inference." In: *arXiv preprint arXiv:1905.06424* (cit. on pp. 20, 39, 55).

Igl, Maximilian, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson (2018). "Deep Variational Reinforcement Learning for POMDPs." In: *International Conference on Machine Learning* (cit. on pp. 39, 40, 54).

Jaderberg, Max, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu (2016). "Reinforcement learning with unsupervised auxiliary tasks." In: *arXiv preprint arXiv:1611.05397* (cit. on p. 6).

James, Stephen, Michael Bloesch, and Andrew J Davison (2018). "Task-Embedded Control Networks for Few-Shot Imitation Learning." In: *Conference on Robot Learning* (cit. on pp. 38, 40).

Jong, Nicholas K and Peter Stone (2005). "State Abstraction Discovery from Irrelevant State Variables." In: *IJCAI*. Vol. 8, pp. 752–757 (cit. on p. 6).

Jonschkowski, Rico and Oliver Brock (2015). "Learning state representations with robotic priors." In: *Autonomous Robots* 39.3, pp. 407–428 (cit. on p. 6).

Kaelbling, Leslie Pack, Michael Littman, and Anthony Cassandra (1998). "Planning and Acting in Partially Observable Stochastic Domains." In: vol. 101, pp. 99–134 (cit. on pp. 39, 40).

Kalashnikov, Dmitry, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. (2018a). "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation." In: *arXiv preprint arXiv:1806.10293* (cit. on p. 1).

Kalashnikov, Dmitry, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. (2018b). "Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation." In: *CoRL*, pp. 651–673 (cit. on p. 4).

Karkus, Peter, David Hsu, and Wee Sun Lee (2017). "Qmdp-net: Deep learning for planning under partial observability." In: *NeurIPS*, pp. 4694–4704 (cit. on p. 40).

Karl, Maximilian, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt (2016). "Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data." In: *ICLR* (cit. on pp. 6, 40).

Kass, Michael, Andrew Witkin, and Demetri Terzopoulos (1988). "Snakes: Active contour models." In: *IJCV* (cit. on p. 24).

Kingma, Diederik P and Max Welling (2014). "Auto-encoding variational bayes." In: *International Conference on Learning Representations* (cit. on pp. 42, 45).

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. 6).

Klyubin, Alexander S, Daniel Polani, and Chrystopher L Nehaniv (2008). "Keep your options open: An information-based driving principle for sensorimotor systems." In: *PloS one* 3.12 (cit. on p. 6).

Kober, Jens, J Andrew Bagnell, and Jan Peters (2013). "Reinforcement learning in robotics: A survey." In: *IJRR* 32.11, pp. 1238–1274 (cit. on p. 40).

Koch, Gregory, Richard Zemel, and Ruslan Salakhutdinov (2015). "Siamese neural networks for one-shot image recognition." In: *ICML Deep Learning Workshop* (cit. on p. 24).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks." In: *NIPS* (cit. on p. 1).

Kronander, Klas, Etienne Burdet, and Aude Billard (2014). "Task transfer via collaborative manipulation for insertion assembly." In: *WHRI*. Citeseer (cit. on p. 40).

Kumar, Visak, Tucker Herman, Dieter Fox, Stan Birchfield, and Jonathan Tremblay (2019). "Contextual Reinforcement Learning of Visuo-tactile Multi-fingered Grasping Policies." In: *arXiv preprint arXiv:1911.09233* (cit. on p. 40).

Lake, Brenden M, Ruslan Salakhutdinov, and Joshua B Tenenbaum (2015). "Human-level concept learning through probabilistic program induction." In: *Science* (cit. on pp. 23, 25).

Lake, Brenden, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum (2011). "One shot learning of simple visual concepts." In: *Proceedings of the annual meeting of the cognitive science society*. Vol. 33. 33 (cit. on p. 3).

Lange, Sascha, Martin Riedmiller, and Arne Voigtländer (2012). "Autonomous reinforcement learning on raw visual input data in a real world application." In: *IJCNN*. IEEE, pp. 1–8 (cit. on p. 40).

Lattimore, Tor and Csaba Szepesvari (2019). "Learning with Good Feature Representations in Bandits and in RL with a Generative Model." In: *arXiv preprint arXiv:1911.07676* (cit. on pp. 6, 9).

Lawrence, Neil D and John C Platt (2004). "Learning to learn with the informative vector machine." In: *Proceedings of the twenty-first international conference on Machine learning*, p. 65 (cit. on p. 3).

Lee, Alex, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine (2019). "Stochastic Latent Actor-Critic: Deep Reinforcement Learning with a Latent Variable Model." In: *arXiv preprint arXiv:1907.00953* (cit. on pp. 4, 6, 8, 37, 40, 54, 57, 61, 95).

Lee, Michelle A, Yuke Zhu, Krishnan Srinivasan, Parth Shah, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Jeannette Bohg (2018). "Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks." In: *arXiv preprint arXiv:1810.10191* (cit. on p. 40).

Leibfried, Felix, Sergio Pascual-Diaz, and Jordi Grau-Moya (2019). "A Unified Bellman Optimality Principle Combining Reward Maximization and Empowerment." In: *Advances in Neural Information Processing Systems*, pp. 7867–7878 (cit. on p. 6).

Lesort, Timothee, Natalia Diaz-Rodriguez, Jean-Franois Goudou, and David Filliat (2018). "State representation learning for control: An overview." In: *Neural Networks* 108, pp. 379–392 (cit. on p. 4).
Levine, Sergey, Chelsea Finn, Trevor Darrell, and Pieter Abbeel (2016). "End-to-end training of deep visuomotor policies." In: *JMLR* 17.1, pp. 1334–1373 (cit. on p. 40).
Li, Lihong, Thomas J Walsh, and Michael L Littman (2006). "Towards a Unified Theory of State Abstraction for MDPs." In: *ISAIM* (cit. on pp. 6, 8, 9).
Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick (2014). "Microsoft COCO: Common objects in context." In: *ECCV*, pp. 740–755 (cit. on p. 34).
Linsker, Ralph (1988). "Self-organization in a perceptual network." In: *Computer* 21.3, pp. 105–117 (cit. on pp. 4, 5).
Liu, C., J. Yuen, and A. Torralba (2011). "Sift flow: Dense correspondence across scenes and its applications." In: *PAMI* (cit. on p. 24).
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. 65).
Maninis, Kevis-Kokitsi, Sergi Caelles, Jordi Pont-Tuset, and Luc Van Gool (2018). "Deep extreme cut: From extreme points to object segmentation." In: *CVPR* (cit. on pp. 24, 27).
McAllester, David and Karl Statos (2018). "Formal limitations on the measurement of mutual information." In: *arXiv preprint arXiv:1811.04251* (cit. on p. 5).
McCallum, Andrew Kachites (1996). "Reinforcement Learning with Selective Perception and Hidden State." Doctoral dissertation. University of Rochester (cit. on p. 6).
Mendonca, Russell, Abhishek Gupta, Rosen Kralev, Pieter Abbeel, Sergey Levine, and Chelsea Finn (2019). "Guided Meta-Policy Search." In: *arXiv preprint arXiv:1904.00956* (cit. on p. 38).
Mirowski, Piotr (2019). "Learning to Navigate." In: *1st International Workshop on Multimodal Understanding and Learning for Embodied Applications*, pp. 25–25 (cit. on p. 6).
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. 38, 45).
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 pp. 4, 40).

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. 6).

Nachum, Ofir, Shixiang Gu, Honglak Lee, and Sergey Levine (2018). "Near-Optimal Representation Learning for Hierarchical Reinforcement Learning." In: (cit. on pp. 6, 8, 9).

Nagabandi, Anusha, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn (2018). "Learning to adapt in dynamic, real-world environments through meta-RL." In: *arXiv preprint arXiv:1803.11347* (cit. on p. 41).

Nagabandi, Anusha, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn (2019). "Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning." In: *International Conference on Learning Representations (ICLR)* (cit. on p. 38).

Newman, Wyatt S, Yonghong Zhao, and Y-H Pao (2001). "Interpretation of force and moment signals for compliant peg-in-hole assembly." In: *ICRA*. Vol. 1. IEEE, pp. 571–576 (cit. on p. 40).

Oord, Aaron van den, Yazhe Li, and Oriol Vinyals (2018). "Representation Learning with Contrastive Predictive Coding." In: *arXiv preprint arXiv:1807.03748* (cit. on pp. 2, 4–6, 8, 10).

Oreshkin, Boris N, Alexandre Lacoste, and Pau Rodriguez (2018). "TADAM: Task dependent adaptive metric for improved few-shot learning." In: (cit. on p. 38).

Osband, Ian, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy (2016). "Deep Exploration via Bootstrapped DQN." In: *Advances in Neural Information Processing Systems* (cit. on p. 44).

Osband, Ian, Benjamin Van Roy, and Daniel Russo (2013). "(More) Efficient Reinforcement Learning via Posterior Sampling." In: *Advances in Neural Information Processing Systems* (cit. on pp. 39, 43).

Pathak, Deepak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell (2017). "Curiosity-driven exploration by self-supervised prediction." In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, pp. 2778–2787 (cit. on pp. 2, 4, 6, 8, 10).

Pereira, Guilherme AS, Mario FM Campos, and Vijay Kumar (2004). "Decentralized algorithms for multi-robot manipulation via caging." In: *IJRR* 23.7-8, pp. 783–795 (cit. on p. 40).

Perez, Christian F, Felipe Petroski Such, and Theofanis Karaletsos (2020). "Generalized Hidden Parameter MDPs Transferable Model-based RL in a Handful of Trials." In: *arXiv preprint arXiv:2002.03072* (cit. on p. 39).

Pineau, Joelle, Geoff Gordon, Sebastian Thrun, et al. (2003). "Point-based value iteration: An anytime algorithm for POMDPs." In: *IJCAI*. Vol. 3, pp. 1025–1032 (cit. on p. 40).

Pont-Tuset, Jordi, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool (2017). "The 2017 DAVIS Challenge on Video Object Segmentation." In: *arXiv:1704.00675* (cit. on pp. 24, 31, 91).

Poole, Ben, Sherjil Ozair, Aaron van den Oord, Alexander A Alemi, and George Tucker (2019). "On variational bounds of mutual information." In: *arXiv preprint arXiv:1905.06922* (cit. on p. 5).

Rakelly, Kate, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine (2019). "Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables." In: *ICML* (cit. on pp. 18, 55, 60, 62, 100, 102).

Learning Speaker Representations with Mutual Information (2019) (cit. on p. 4).

Ravi, Sachin and Hugo Larochelle (2017). "Optimization as a model for few-shot learning." In: *ICLR* (cit. on pp. 2, 18, 24, 38).

Ravindran, Balaraman and Andrew G Barto (2003). "SMDP homomorphisms: an algebraic approach to abstraction in semi-Markov decision processes." In: *Proceedings of the 18th international joint conference on Artificial intelligence*, pp. 1011–1016 (cit. on p. 6).

Ren, Hongyu, Yuke Zhu, Jure Leskovec, Animashree Anandkumar, and Animesh Garg (2020). "OCEAN: Online Task Inference for Compositional Tasks with Context Adaptation." In: *Conference on Uncertainty in Artificial Intelligence*. PMLR, pp. 1378–1387 (cit. on p. 18).

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). "Stochastic Backpropagation and Approximate Inference in Deep Generative Models." In: *International Conference on Machine Learning* (cit. on p. 42).

Ross, Stéphane, Joelle Pineau, Brahim Chaib-draa, and Pierre Kreitmann (2011). "A Bayesian approach for learning and planning in partially observable Markov decision processes." In: *JMLR* 12.May, pp. 1729–1770 (cit. on p. 40).

Rothfuss, Jonas, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel (2018). "ProMP: Proximal Meta-Policy Search." In: *International Conference on Learning Representations* (cit. on pp. 36, 38, 48).

Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. (2015). "Im-

agenet large scale visual recognition challenge." In: *International Journal of Computer Vision* 115.3, pp. 211–252 (cit. on pp. 31, 92).

Rusu, Andrei A, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell (2019). "Meta-learning with latent embedding optimization." In: *International Conference on Learning Representations* (cit. on pp. 36, 39).

Sæmundsson, Steindór, Katja Hofmann, and Marc Peter Deisenroth (2018). "Meta Reinforcement Learning with Latent Variable Gaussian Processes." In: *Conference on Uncertainty in Artificial Intelligence (UAI)* (cit. on p. 38).

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. 2, 18, 24, 38).

Sax, Alexander, Bradley Emi, Amir R. Zamir, Leonidas J. Guibas, Silvio Savarese, and Jitendra Malik (2019). "Mid-Level Visual Representations Improve Generalization and Sample Efficiency for Learning Visuomotor Policies." In: *CoRL* (cit. on p. 40).

Schaul, Tom, Diana Borsa, Joseph Modayil, and Razvan Pascanu (2019). "Ray Interference: a Source of Plateaus in Deep Reinforcement Learning." In: *arXiv preprint arXiv:1904.11455* (cit. on p. 67).

Schmidhuber, Jürgen (1987). "Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook." Doctoral dissertation. Technische Universität München (cit. on pp. 2, 37).

Schmidt, Tanner, Richard Newcombe, and Dieter Fox (2016). "Self-supervised visual descriptor learning for dense correspondence." In: *IEEE Robotics and Automation Letters* 2.2, pp. 420–427 (cit. on p. 40).

Schoettler, Gerrit, Ashvin Nair, Jianlan Luo, Shikhar Bahl, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine (2019). "Deep Reinforcement Learning for Industrial Insertion Tasks with Visual Inputs and Natural Reward Signals." In: *ICLR* (cit. on p. 40).

Schulman, John, Filip Wolski, Prafulla Dhariwal Dhariwal, Alec Radford Radford, and Oleg Klimov (2017). "Proximal Policy Optimization Algorithms." In: *arXiv:1707.06347* (cit. on p. 48).

Shaban, Amirreza, Shray Bansal, Zhen Liu, Irfan Essa, and Byron Boots (2017). "One-Shot Learning for Semantic Segmentation." In: *BMVC* (cit. on pp. 24, 27, 28, 33, 91).

Shelhamer, Evan, Jonathan Long, and Trevor Darrell (2016a). "Fully Convolutional Networks for Semantic Segmentation." In: *PAMI* (cit. on pp. 1, 31, 92).

Shelhamer, Evan, Parsa Mahmoudieh, Max Argus, and Trevor Darrell (2016b). "Loss is its own reward: Self-supervision for reinforcement learning." In: *arXiv preprint arXiv:1612.07307* (cit. on pp. 4, 6, 10).

Shinners, Pete (2011). *PyGame*. http://pygame.org/ (cit. on p. 13).

Simonyan, Karen and Andrew Zisserman (2015). "Very Deep Convolutional Networks for Large-Scale Image Recognition." In: *ICLR* (cit. on pp. 31, 92).

Singh, Avi, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine (2020). "End-to-End Robotic Reinforcement Learning without Reward Engineering." In: *environment (eg, by placing additional sensors)* 34, p. 44 (cit. on p. 40).

Snell, Jake, Kevin Swersky, and Richard Zemel (2017). "Prototypical networks for few-shot learning." In: *NIPS* (cit. on pp. 18, 24, 38).

Song, Jiaming and Stefano Ermon (2019). "Understanding the Limitations of Variational Mutual Information Estimators." In: *arXiv preprint arXiv:1910.06222* (cit. on pp. 4, 5).

Song, Xingyou, Yuxiang Yang, Krzysztof Choromanski, Ken Caluwaerts, Wenbo Gao, Chelsea Finn, and Jie Tan (2020). "Rapidly Adaptable Legged Robots via Evolutionary Meta-Learning." In: *arXiv preprint arXiv:2003.01239* (cit. on p. 40).

Stadie, Bradly C, Ge Yang, Rein Houthooft, Xi Chen, Yan Duan, Yuhuai Wu, Pieter Abbeel, and Ilya Sutskever (2018). "Some considerations on learning to explore via meta-reinforcement learning." In: *arXiv preprint arXiv:1803.01118* (cit. on p. 38).

Still, Susanne and Doina Precup (2012). "An information-theoretic approach to curiosity-driven reinforcement learning." In: *Theory in Biosciences* 131.3, pp. 139–148 (cit. on p. 6).

Stooke, Adam, Kimin Lee, Pieter Abbeel, and Michael Laskin (2020). *Decoupling Representation Learning from Reinforcement Learning*. Tech. rep. UC Berkeley (cit. on pp. 6, 10).

Strens, Malcom (2000). "A Bayesian Framework for Reinforcement Learning." In: *International Conference on Machine Learning* (cit. on pp. 39, 43).

Sun, Chen, Fabien Baradel, Kevin Murphy, and Cordelia Schmid (2019). "Contrastive bidirectional transformer for temporal representation learning." In: *arXiv preprint arXiv:1906.05743* (cit. on p. 5).

Sung, Flood, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales (2018). "Learning to compare: Relation network for few-shot learning." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1199–1208 (cit. on p. 18).

Sung, Flood, Li Zhang, Tao Xiang, Timothy Hospedales, and Yongxin Yang (2017). "Learning to learn: Meta-critic networks for sample efficient learning." In: *arXiv preprint arXiv:1706.09529* (cit. on p. 38).

Sutton, Richard S and Andrew G Barto (2018). *Reinforcement learning: An introduction*. MIT press (cit. on p. 9).

Sutton, Richard S, Andrew G Barto, Francis Bach, et al. (1998). *Reinforcement learning: An introduction*. MIT press (cit. on p. 19).

Tenenbaum, Joshua Brett (1999). "A Bayesian framework for concept learning." Doctoral dissertation. Massachusetts Institute of Technology (cit. on pp. 3, 39).

Thrun, Sebastian and Lorien Pratt (1998). *Learning to learn*. Springer Science & Business Media (cit. on pp. 2, 17, 37).

Todorov, Emanuel, Tom Erez, and Yuval Tassa (2012). "MuJoCo: A physics engine for model-based control." In: *International Conference on Intelligent Robots and Systems (IROS)*, pp. 5026–5033 (cit. on pp. 48, 60).

Tremblay, Jonathan, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield (2018). "Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects." In: *CoRL*, pp. 306–316 (cit. on p. 40).

Van Roy, Benjamin and Shi Dong (2019). "Comments on the Du-Kakade-Wang-Yang Lower Bounds." In: *arXiv preprint arXiv:1911.07910* (cit. on p. 6).

Veeriah, Vivek, Matteo Hessel, Zhongwen Xu, Janarthanan Rajendran, Richard L Lewis, Junhyuk Oh, Hado P van Hasselt, David Silver, and Satinder Singh (2019). "Discovery of useful questions as auxiliary tasks." In: *Advances in Neural Information Processing Systems*, pp. 9306–9317 (cit. on p. 6).

Vinyals, Oriol, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. (2019). "Grandmaster level in StarCraft II using multi-agent reinforcement learning." In: *Nature* 575.7782, pp. 350–354 (cit. on p. 1).

Vinyals, Oriol, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. (2016). "Matching networks for one shot learning." In: *NIPS* (cit. on pp. 17, 18, 24, 25, 35, 38).

Wainwright, Martin J and Michael Irwin Jordan (2008). *Graphical models, exponential families, and variational inference*. Now Publishers Inc (cit. on p. 57).

Wang, Yu-Xiong and Martial Hebert (2016). "Learning to learn: Model regression networks for easy small sample learning." In: *ECCV* (cit. on pp. 2, 20, 24, 36, 38, 55).

Watter, Manuel, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller (2015). "Embed to control: A locally linear latent dynamics model for control from raw images." In: *NeurIPS*, pp. 2746–2754 (cit. on pp. 2, 6, 8, 40).

Xie, Annie, Avi Singh, Sergey Levine, and Chelsea Finn (2018). "Few-Shot Goal Inference for Visuomotor Learning and Planning." In: *CoRL*, pp. 40–52 (cit. on p. 40).

Xu, Ning, Brian Price, Scott Cohen, Jimei Yang, and Thomas S Huang (2016). "Deep interactive object selection." In: *CVPR*, pp. 373–381 (cit. on pp. 24, 27, 31, 91).

Xu, Tianbing, Qiang Liu, Liang Zhao, and Jian Peng (2018). "Learning to explore via meta-policy gradient." In: *International Conference on Machine Learning*, pp. 5459–5468 (cit. on p. 38).

Xu, Zhongwen, Hado van Hasselt, and David Silver (2018). "Meta-Gradient Reinforcement Learning." In: *arXiv:1805.09801* (cit. on p. 38).

Yadaiah, Narri and G Sowmya (2006). "Neural network based state estimation of dynamical systems." In: *IJCNN*. IEEE, pp. 1042–1049 (cit. on p. 40).

Yarats, Denis, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus (2019). "Improving Sample Efficiency in Model-Free Reinforcement Learning from Images." In: *arXiv preprint arXiv:1910.01741* (cit. on p. 40).

Yoon, Jae Shin, Francois Rameau, Junsik Kim, Seokju Lee, Seunghak Shin, and In So Kweon (2017). "Pixel-Level Matching for Video Object Segmentation using Convolutional Neural Networks." In: *ICCV*, pp. 2186–2195 (cit. on p. 27).

Yoon, Jaesik, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn (2018). "Bayesian model-agnostic meta-learning." In: *Advances in Neural Information Processing Systems*, pp. 7343–7353 (cit. on p. 39).

Yu, Tianhe, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine (2018). "One-Shot Imitation from Observing Humans via Domain-Adaptive Meta-Learning." In: *ICLR* (cit. on p. 40).

Yu, Tianhe, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn (2020). "Gradient surgery for multi-task learning." In: *arXiv preprint arXiv:2001.06782* (cit. on p. 67).

Yu, Tianhe, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine (2019a). "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning." In: *arXiv preprint arXiv:1910.10897* (cit. on p. 67).

Yu, Tianhe, Gleb Shevchuk, Dorsa Sadigh, and Chelsea Finn (2019b). "Unsupervised Visuomotor Control through Distributional Planning Networks." In: *Robotics Science and Systems* (cit. on pp. 6, 10).

Zeng, Andy, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser (2018). "Learning Synergies between Pushing and Grasping with Self-supervised Deep RL." In: *arXiv preprint arXiv:1803.09956* (cit. on p. 40).

Zhang, Amy, Harsh Satija, and Joelle Pineau (2018). "Decoupling dynamics and reward for transfer learning." In: *arXiv preprint arXiv:1804.10689* (cit. on pp. 6, 10).

Zhang, Marvin, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew J Johnson, and Sergey Levine (2018). "Solar: Deep structured latent representations for model-based reinforcement learning." In: *arXiv preprint arXiv:1808.09105* (cit. on p. 6).

Zhang, Marvin, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew Johnson, and Sergey Levine (2019). "SOLAR: Deep Structured Representations for Model-Based Reinforcement Learning." In: *ICML*, pp. 7444–7453 (cit. on pp. 37, 40).

Zhu, Henry, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey Levine (2020). "The Ingredients of Real-World Robotic Reinforcement Learning." In: *arXiv preprint arXiv:2004.12570* (cit. on p. 65).

Zintgraf, Luisa, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson (2019). "VariBAD: A Very Good Method for Bayes-Adaptive Deep RL via Meta-Learning." In: *arXiv preprint arXiv:1910.08348* (cit. on pp. 20, 39, 55).

A

APPENDIX: WHAT MUTUAL INFORMATION-BASED REPRESENTATION LEARNING OBJECTIVES ARE SUFFICIENT FOR CONTROL?

This appendix accompanies the work presented in Chapter 2.

A.1 SUFFICIENCY OF $\mathbb{J}_{fwd}$: PROOF OF PROPOSITION 1

We describe the proofs for the sufficiency results from Section 2.4 here. We begin by providing a set of lemmas, before proving the sufficiency of $\mathbb{J}_{fwd}$.

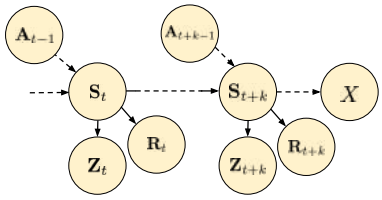


Figure 29: Graphical model for Lemma 1, depicting true states S, states in the representation Z, actions A, rewards R, and the variable X (which we will interpret as the sum of future rewards in the proof of Proposition 1).

Lemma 1. *Let X be a random variable dependent on S_{t+k}, with the conditional independence assumptions implied by the graphical model in Figure 29. (In the main proof of Proposition 1, we will let X be the sum of rewards from time $t+k$ onwards.) If $I(Z_{t+k}; Z_t, A_t) = I(S_{t+k}; S_t, A_t) \forall k$, then $I(X; Z_t, A_t) = I(X; S_t, A_t) \forall k$.*

Proof. For proof by contradiction, assume there is some ϕ_Z and some r such that $I(X; Z_t, A_t) < I(X; S_t, A_t)$ and that $I(Z_{t+k}; Z_t, A_t) = I(S_{t+k}; S_t, A_t)$. Now we know that because $Z_t \rightarrow$

$S_t \to S_{t+k} \to Z_{t+k}$ form a Markov chain, by the data processing inequality (DPI) $I(Z_{t+k}; Z_t, A_t) \leqslant I(S_{t+k}; Z_t, A_t) \leqslant I(S_{t+k}; S_t, A_t)$. We will proceed by showing that that $I(X; Z_t, A_t) < I(X; S_t, A_t) \implies I(S_{t+k}; Z_t, A_t) < I(S_{t+k}; S_t, A_t)$, which gives the needed contradiction.

Using chain rule, we can expand the following expression in two different ways.

$$I(X; Z_t, S_t, A_t) = I(X; Z_t | S_t, A_t) + I(X; S_t, A_t) = 0 + I(X; S_t, A_t) \tag{16}$$

$$I(X; Z_t, S_t, A_t) = I(X; S_t | Z_t, A_t) + I(X; Z_t, A_t) \tag{17}$$

Note that the first term in Equation 16 is zero by the conditional independence assumptions in Figure 29. Equating the expansions, we can see that to satisfy our assumption that $I(X; Z_t, A_t) < I(X; S_t, A_t)$, we must have that $I(X; S_t | Z_t, A_t) > 0$.

Now we follow a similar procedure to expand the following expression:

$$I(S_{t+k}; Z_t, S_t, A_t) = I(S_{t+k}; Z_t | S_t, A_t) + I(S_{t+k}; S_t, A_t) = 0 + I(S_{t+k}; S_t, A_t) \tag{18}$$

$$I(S_{t+k}; Z_t, S_t, A_t) = I(S_{t+k}; S_t | Z_t, A_t) + I(S_{t+k}; Z_t, A_t) \tag{19}$$

The first term in Equation 18 is zero by the conditional independence assumptions in Figure 29. Comparing the first term in Equation 19 with the first term in Equation 17, we see because $S_t \to S_{t+k} \to X$ form a Markov chain, by the DPI that $I(S_{t+k}; S_t | Z_t, A_t) \geqslant I(X; S_t | Z_t, A_t)$. Therefore we must have $I(S_{t+k}; S_t | Z_t, A_t) > 0$. Combining Equations 18 and 19:

$$I(S_{t+k}; S_t, A_t) = I(S_{t+k}; S_t | Z_t, A_t) + I(S_{t+k}; Z_t, A_t) \tag{20}$$

Since $I(S_{t+k}; S_t | Z_t, A_t) > 0$, $I(S_{t+k}; Z_t, A_t) < I(S_{t+k}; S_t, A_t)$, which is exactly the contradiction we set out to show. □

Lemma 2. *If* $I(Y; Z) = I(Y; X)$ *and* $Y \perp Z | X$, *then* $\exists p(Z|X)$ *s.t.* $\forall x$, $p(Y|X = x) = \int p(Y|Z) p(Z|X = x) dz$.

Proof. First note that the statement is not trivially true. Without any assumption regarding MI, we can write,

$$p(Y|X = x) = \int p(Y, Z|X = x) dz = \int p(Y|Z, X = x) p(Z|X = x) dz \tag{21}$$

Comparing this with the statement we'd like to prove, we can see that the key idea is to show that the MI equivalence implies that $p(Y|Z, X = x) = p(Y|Z)$. To begin, consider $I(Y; Z) = I(Y; X)$. We can re-write this equality using the entropy definition of MI.

$$H(Y) - H(Y|Z) = H(Y) - H(Y|X) \tag{22}$$

Note that the $H(Y)$ cancel and substituting the definition of entropy we have:

$$\mathbb{E}_{p(Y,Z)}[\log p(Y|Z)] = \mathbb{E}_{p(Y,X)}[\log p(Y|X)] \tag{23}$$

Note that on the right-hand side, we can use the Tower property to re-write the expectation as

$$\mathbb{E}_{p(Y,X)}[\log p(Y|X)] = \mathbb{E}_{p(Z)}\mathbb{E}_{p(Y,X|Z)}[\log p(Y|X)] = \mathbb{E}_{p(Y|X)p(X,Z)}[\log p(Y|X)] \tag{24}$$

Now we can use the Tower property again to re-write the expectation on both sides.

$$\begin{aligned}
\mathbb{E}_{p(X)}[\mathbb{E}_{p(Y,Z|X)}[\log p(Y|Z)]] &= \mathbb{E}_{p(X)}[\mathbb{E}_{p(Y|X)p(X,Z|X)}[\log p(Y|X)]] \\
\mathbb{E}_{p(X)}[\mathbb{E}_{p(Y|X)p(Z|X)}[\log p(Y|Z)]] &= \mathbb{E}_{p(X)}[\mathbb{E}_{p(Y|X)p(Z|X)}[\log p(Y|X)]] \\
\mathbb{E}_{p(X)}[\mathbb{E}_{p(Y|X)p(Z|X)}[\log p(Y|Z)]] - \log p(Y|X)] &= 0 \\
\mathbb{E}_{p(X)}[\mathbb{E}_{p(Y|X)}[\mathbb{E}_{p(Z|X)}[\log p(Y|Z)] - \log p(Y|X)]] &= 0
\end{aligned} \tag{25}$$

Log probabilities are always $\leqslant 0$, therefore for the sum to equal zero, each term must be zero.

$$\log p(Y|X) = \mathbb{E}_{p(Z|X)}[\log p(Y|Z)] \tag{26}$$

By Jensen's inequality,

$$\log p(Y|X) \leqslant \log \mathbb{E}_{p(Z|X)}[p(Y|Z)] \tag{27}$$

By the monotonicity of the logarithm:

$$p(Y|X) \leqslant \mathbb{E}_{p(Z|X)}[p(Y|Z)] \tag{28}$$

If there exists some x and some y such that $p(Y = y|X = x) < \mathbb{E}_{p(Z|X=x)}[p(Y = y|Z)]$, then there must be some other y' for the same x where $p(Y = y'|X = x) > \mathbb{E}_{p(Z|X=x)}[p(Y = y'|Z)]$ because $p(Y|X = x)$ must sum to 1.

$$p(Y|X) = \mathbb{E}_{p(Z|X)}[p(Y|Z)] = \int p(Y|Z)p(Z|X = x)dz = \int p(Y, Z|X = x)dz \tag{29}$$

Where the last equality follows by conditional independence of Y and Z given X. □

Given the lemmas stated above, we can then use them to prove the sufficiency of $\mathbb{J}_{fwd}$.

Proposition 1. *(Sufficiency of $\mathbb{J}_{fwd}$) Let $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r)$ be an MDP with dynamics $p(S_{t+1}|S_t, A_t)$. Let the policy distribution $p(A|S)$ and steady-state state occupancy $p(S)$ have full support on the action and state alphabets $\mathcal{A}$ and $\mathcal{S}$ respectively. See Figure 29 for a graphical depiction of the conditional independence relationships between variables.*

For a representation ϕ_Z and set of reward functions $\mathcal{R}$, if $I(Z_{t+k}; Z_t, A_t)$ is maximized $\forall k > 0, t > 0$ then $\forall r \in \mathcal{R}$ and $\forall \mathbf{s}_1, \mathbf{s}_2 \in \mathcal{S}$, $\phi_Z(\mathbf{s}_1) = \phi_Z(\mathbf{s}_2) \implies \forall \mathbf{a}, Q^*_r(\mathbf{a}, \mathbf{s}_1) = Q^*_r(\mathbf{a}, \mathbf{s}_2)$.

Proof. Note that $(Z_{t+k}; Z_t, A_t)$ is maximized if the representation ϕ_Z is taken to be the identity. In other words $\max_\phi I(Z_{t+k}; Z_t, A_t) = I(S_{t+k}; S_t, A_t)$.

Define the random variable $\bar{R}_t$ to be the discounted return starting from state $\mathbf{s}_t$.

$$\bar{R}_t = \sum_{k=1}^{H-t} \gamma^k R_{t+k} \tag{30}$$

Plug in $\bar{R}_t$ for the random variable X in Lemma 1:

$$I(Z_{t+k}; Z_t, A_t) = I(S_{t+k}; S_t, A_t) \qquad \implies \qquad I(\bar{R}_{t+k}; Z_t, A_t) = I(\bar{R}_{t+k}; S_t, A_t) \tag{31}$$

Now let $X = [S_t, A_t]$, $Y = \bar{R}_t$, and $Z = Z_t$, and note that by the structure of the graphical model in Figure 29, $Y \perp Z|X$. Plugging into Lemma 2:

$$\mathbb{E}_{p(\mathbf{z}_t|S_t=\mathbf{s})} p(\bar{R}_t|Z_t, A_t) = p(\bar{R}_t|S_t = \mathbf{s}, A_t) \tag{32}$$

Now the Q-function given a reward function r and a state-action pair $(\mathbf{s}, \mathbf{a})$ can be written as an expectation of this random variable $\bar{R}_t$, given $S_t = \mathbf{s}$ and $A = \mathbf{a}$. (Note that $p(\bar{R}_t|S_t = \mathbf{s}, A_t = \mathbf{a})$ can be calculated from the dynamics, policy, and reward distributions.)

$$Q_r(\mathbf{s}, \mathbf{a}) = \mathbb{E}_{p(\bar{R}_t|S_t=\mathbf{s}, A_t=\mathbf{a})}[\bar{R}_t] \tag{33}$$

Since $\phi_Z(\mathbf{s}_1) = \phi_Z(\mathbf{s}_2)$, $p(\mathbf{z}_t|S_t = \mathbf{s}_1) = p(\mathbf{z}_t|S_t = \mathbf{s}_2)$. Therefore by Equation 32, $p(\bar{R}_t|S_t = \mathbf{s}_1, A_t) = p(\bar{R}_t|S_t = \mathbf{s}_2, A_t)$. Plugging this result into Equation 33, $Q_r(\mathbf{a}, \mathbf{s}_1) = Q_r(\mathbf{a}, \mathbf{s}_2)$. Because this reasoning holds for all Q-functions [1], it also holds for the optimal Q, therefore $Q^*_r(\mathbf{a}, \mathbf{s}_1) = Q^*_r(\mathbf{a}, \mathbf{s}_2)$.

□

1 Note this result is stronger than what we needed: it means that representations that maximize $\mathbb{J}_{fwd}$ are guaranteed to be able to represent even sub-optimal Q-functions. This makes sense in light of the fact that the proof holds for all reward functions - the sub-optimal Q under one reward is the optimal Q under another.

A.2 EXPERIMENTAL DETAILS

A.2.0.1 *Didactic Experiments*

The didactic examples are computed as follows. Given the list of states in the MDP, we compute the possible representations, restricting our focus to representations that group states into "blocks." We do this because there are infinite stochastic representations and the MI expressions we consider are not convex in the parameters of $p(Z|S)$, making searching over these representations difficult. Given each state representation, we compute the value of the MI objective as well as the optimal value function using exact value iteration. In these examples, we assume that the policy distribution is uniform, and that the environment dynamics are deterministic. Since we consider the infinite horizon setting, we use the steady-state state occupancy in our calculations.

A.2.0.2 *Deep RL Experiments*

The deep RL experiments with the catcher game are conducted as follows. First, we use a uniform random policy to collect 50k transitions in the environment. In this simple environment, the uniform random policy suffices to visit all states (the random agent is capable of accidentally catching the fruit, for example). Next, each representation learning objective is maximized on this dataset. For all objectives, the images are preprocessed in the same manner (resized to 64x64 pixels and normalized) and embedded with a convolutional network. The convolutional encoder consists of five convolutional layers with ReLU activations and produces a latent vector with dimension 256. We use the latent vector to estimate each mutual information objective, as described below.

Inverse information: We interpret the latent embeddings of the images S_t and S_{t+1} as the parameters of Gaussian distributions $p(Z|S_t)$ and $p(Z|S_{t+1})$. We obtain a single sample from each of these two distributions, concatenate them and pass them through a single linear layer to predict the action. The objective we maximize is the cross-entropy of the predicted actions with the true actions, as in Agrawal et al. 2016 and Shelhamer et al. 2016. To prevent recovering the trivial solution of preserving all the information in the image, we add an information bottleneck to the image embeddings. We tune the Lagrange multiplier on this bottleneck such that the action prediction loss remains the same value as when trained without the bottleneck. This approximates the objective $\min_\phi I(Z;S) s.t. I_{inv} = \max I_{inv}$. To use the learned encoder for RL, we embed the image from the current timestep and take the mean of the predicted distribution as the state for the RL agent.

State-only information: We follow the Noise Contrastive Estimation (NCE) approach presented in CPC (Oord et al. 2018). Denoting Z_t and $Zt+1$ as the latent embedding vectors from the convolutional encoders, we use a log-bilinear model as in CPC to compute the score: $f(Z_t, Z_{t+1}) = \exp(Z_t^T W Z_{t+1})$ for the cross-entropy loss. We also experimented with an information bottleneck as described above, but found that it wasn't needed to obtain insufficient representations. To use the learned encoder for RL, we embed the image from the current timestep and use this latent vector as the state for the RL agent.

Forward information: We follow the same NCE strategy as for state-only information, with the difference that we concatenate the action to Z_t before computing the contrastive loss.

We then freeze the state encoder learned via MI-maximization and use the representation as the state input for RL. The RL agent is trained using the Soft Actor-Critic algorithm Haarnoja et al., 2018, modified slightly for the discrete action distribution (the Q-function outputs Q-values for all actions rather than taking action as input, the policy outputs the action distribution rather than parameters of a distribution, and we can directly compute the expectation in the critic loss rather than sampling). The policy and critic networks consist of two hidden linear layers of 200 units each. We use ReLU activations.

A.2.1 *Analysis: predicting* Q^* *from the representation*

In Section 2.5, we evaluated the learned representations by running a temporal difference RL algorithm with the representation as the state input. In this section, instead of using the bootstrap to learn the Q-function, we instead regress the Q-function to the optimal Q^*. To do this, we first compute the (roughly) optimal Q^* by running RL with ground truth game state as input and taking the learned Q as Q^*. Then, we instantiate a new RL agent and train it with the learned image representation as input, regressing the Q-function directly onto the values of Q^*. We evaluate the policy derived from this new Q-function, and plot the results for both the *catcher* and *catcher-grip* environments in Figure 30. We find that similar to the result achieved using the bootstrap, the policy performs poorly when using representations learned by insufficient objectives ($\mathbb{J}_{inv}$ in *catcher* and $\mathbb{J}_{state}$ in *catcher-grip*). Interestingly, we find that the error between the learned Q-values and the Q^*-values is roughly the same for sufficient and insufficient representations. We hypothesize that this discrepancy between Q-value error and policy performance is due to the fact that small differences in Q-values on a small set of states can result in significant behavior differences in the policy.

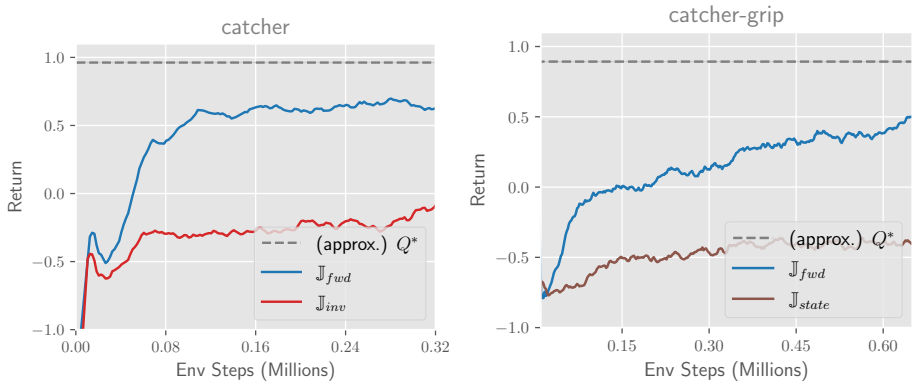


Figure 30: Performance of policies obtained from a Q-function trained to predict Q^*, given state representations learned by each MI objective, in the (left) *catcher* environment and (right) *catcher-grip* environment. Insufficient objectives $\mathbb{J}_{\mathtt{inv}}$ and $\mathbb{J}_{\mathtt{state}}$ respectively perform worse than sufficient objective $\mathbb{J}_{\mathtt{fwd}}$.

A.2.2 *Deep RL experiments with background distractors*

In this section we repeat the experiments from Section 2.5 with added visual complexity in the form of background distractors. We randomly generate images of 10 circles of different colors and replace the black background of the game with these images. Examples of the agent's observations are shown in Figure 31.

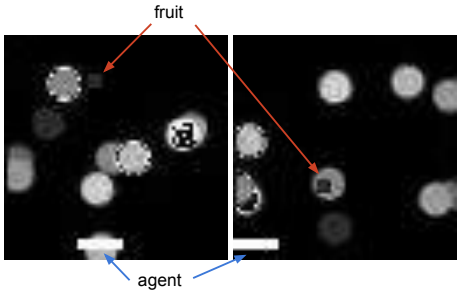
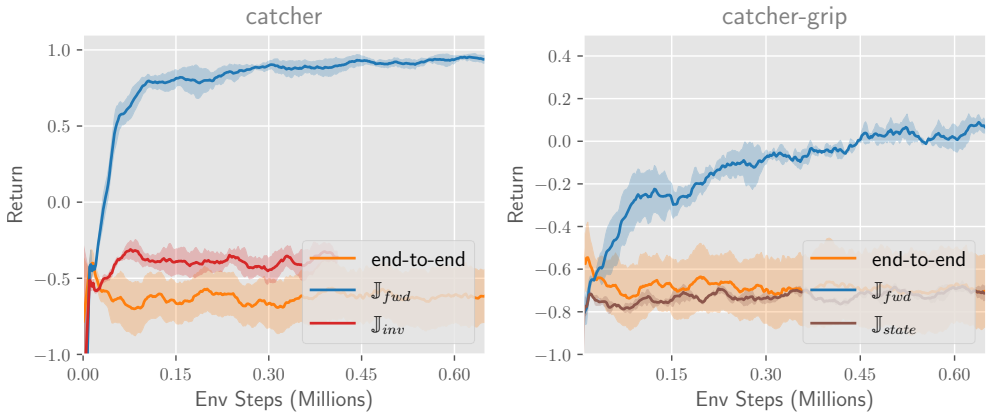


Figure 31: Example 64x64 pixel observations with background distractors.

We plot the results for both the *catcher* and *catcher-grip* games with distractors in Figure 32. As in Section 2.5, we show both the result of performing RL with the frozen representation as input (top), as well as the error of decoding true state elements from the representation (bottom). In both environments, end-to-end RL from images performs poorly, demonstrating the need for representation learning to aid in solving the task. As predicted by the theory, the representation learned by $\mathbb{J}_{\mathtt{inv}}$ fails in both games, and the representation learned by $\mathbb{J}_{\mathtt{state}}$ fails in the *catcher-grip* game. We find that the difference in performance between sufficient and insufficient objectives is even more pronounced in this setting than in the plain background setting.



Obj.	Agent Err.	Fruit Err.
$\mathbb{J}_{\text{fwd}}$	0.1	0.1
$\mathbb{J}_{\text{inv}}$	0.15	0.47

Obj.	Agent Err.	Fruit Err.	Grip Err.
$\mathbb{J}_{\text{fwd}}$	0.1	0.1	0.3
$\mathbb{J}_{\text{state}}$	0.1	0.1	0.5

Figure 32: (top) Policy performance using learned representations as state inputs to RL, for the *catcher* and *catcher-grip* environments with background distractors. (bottom) Error in predicting the positions of ground truth state elements from each learned representation. Representations maximizing $\mathbb{J}_{\text{inv}}$ need not represent the fruit, while representations maximizing $\mathbb{J}_{\text{state}}$ need not represent the gripper, leading these representations to perform poorly in *catcher* and *catcher-grip* respectively.

B

APPENDIX: GUIDED SEGMENTATION VIA META-LEARNING

This appendix accompanies the work presented in Chapter 4.

B.1 DATA PREPARATION

Semantic Segmentation and Interactive Segmentation on PASCAL We use PASCAL VOC 2012 (Everingham et al., 2010) with the additional annotations of SBDD (Hariharan et al., 2011). We define the training set to be the union of the VOC and SBDD training sets, and take the validation set to be the union of VOC and SBDD validation sets, excluding the images in VOC val that overlap with SBDD train. We sparsify the dense masks with random sampling, which we found resulted in performance about equal to the more complex sampling strategies of N. Xu et al. (2016). Thus for a given P, we sample P points randomly from each of the objects to be segmented, as well as the background. Labels for classes or instances that are not part of the task are relabeled to background. The process of sampling a task and sparsifying and remapping the ground truth labels is illustrated in Fig. 9.

For few-shot semantic segmentation, we follow the experimental protocol of Shaban et al. (2017). We test few-shot performance on held-out classes by dividing the 20 classes of PASCAL into 4 sets of 5 classes. Images that contain both held-out and training classes are placed in the held-out set. We subsample splits with more images to ensure that each split contains the same number of images. For each split, we meta-train a guided segmentor with binary tasks sampled from the 15 training classes. We then compute the average performance across 1000 binary tasks sampled from the 5 held-out classes, and average across all four splits.

Video Object Segmentation on DAVIS 2017 We use the DAVIS 2017 benchmark (Pont-Tuset et al., 2017) of 2-3s video clips. We meta-train on the training videos and report average performance on the validation videos. During training, we synthesize tasks

by sampling any two frames from the same video and treating one as the support and the other as the query. During testing, the support consists of all labeled frames, while the remaining frames comprise the query. For the video object segmentation benchmark, the first frame is densely labeled. For interactive video segmentation, varying numbers of frames are labeled with varying numbers of pixelwise labels.

B.1.1 *Architecture and Optimization*

The backbone of our guided networks as well as our baseline networks is VGG-16 (Simonyan and Zisserman, 2015), pre-trained on ILSVRC Russakovsky et al., 2015, and cast into fully convolutional form (Shelhamer et al., 2016a). We largely follow the optimization procedure for FCNs detailed in (Shelhamer et al., 2016a): we optimize our guided nets by SGD with a learning rate of $1e^{-5}$, batch size 1, high momentum 0.99, and weight decay of $5e^{-4}$. The interpolation weights in the decoder are fixed to bilinear and not learned. Note that we normalize the loss by the number of pixels in each image in order to simplify learning rate selection across different datasets with varying image dimensions.

B.1.2 *Metric*

Intersection-over-union (IU) is a standard metric for segmentation, but different families of segmentation tasks choose different forms of the metric. We report the IU of positives averaged across all tasks and masks, defined as $\frac{\sum_i tp_i}{\sum_i tp_i+fp_i+fn_i}$ where i ranges over ground truth segment masks. This choice makes performance comparable across tasks, because it is independent of the number of classes. We choose not to include negatives in the metric because it adds no information, given the binary nature of the scoring, even for multi-class predictions and ground truth since these are handled as a set of binary tasks by the metric. Note that this metric is not directly comparable to the mean IU across classes typically reported for semantic segmentation benchmarks. As a point of comparison, the 0.62 mean IU achieved by FCN-32s on the PASCAL segmentation benchmark corresponds to 0.45 positive IU.

C

APPENDIX: EFFICIENT OFF-POLICY META-RL VIA PROBABILISTIC CONTEXT VARIABLES

This appendix accompanies the work presented in Chapter 5, Section 5.2.

C.1 EXPERIMENTAL DETAILS

Here we provide further details for the MuJoCo continuous control domains in Section 5.2.3. The agents used in these domains are visualized in Figure 33. The horizon in all tasks is 200 steps.

- Half-Cheetah-Dir: move forward and backward (2 tasks)
- Half-Cheetah-Vel: achieve a target velocity running forward (100 train tasks, 30 test tasks)
- Humanoid-Dir-2D: run in a target direction on 2D grid (100 train tasks, 30 test tasks)
- Ant-Fwd-Back: move forward and backward (2 tasks)
- Ant-Goal-2D: navigate to a target goal location on 2D grid (100 train tasks, 30 test tasks)
- Walker-2D-Params: agent is initialized with some system dynamics parameters randomized and must move forward (40 train tasks, 10 test tasks)

For these domains, the on-policy baseline approaches require many more samples to learn the benchmark tasks. Here in Figure 34 we plot the same data as in Figure 14 for the full number of time steps used by the baselines.

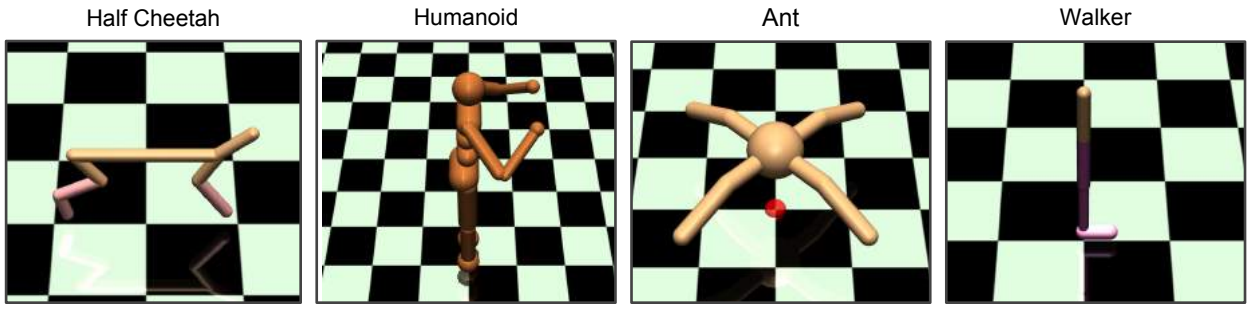


Figure 33: **Continuous control tasks**: left-to-right: the half-cheetah, humanoid, ant, and walker robots used in our evaluation.

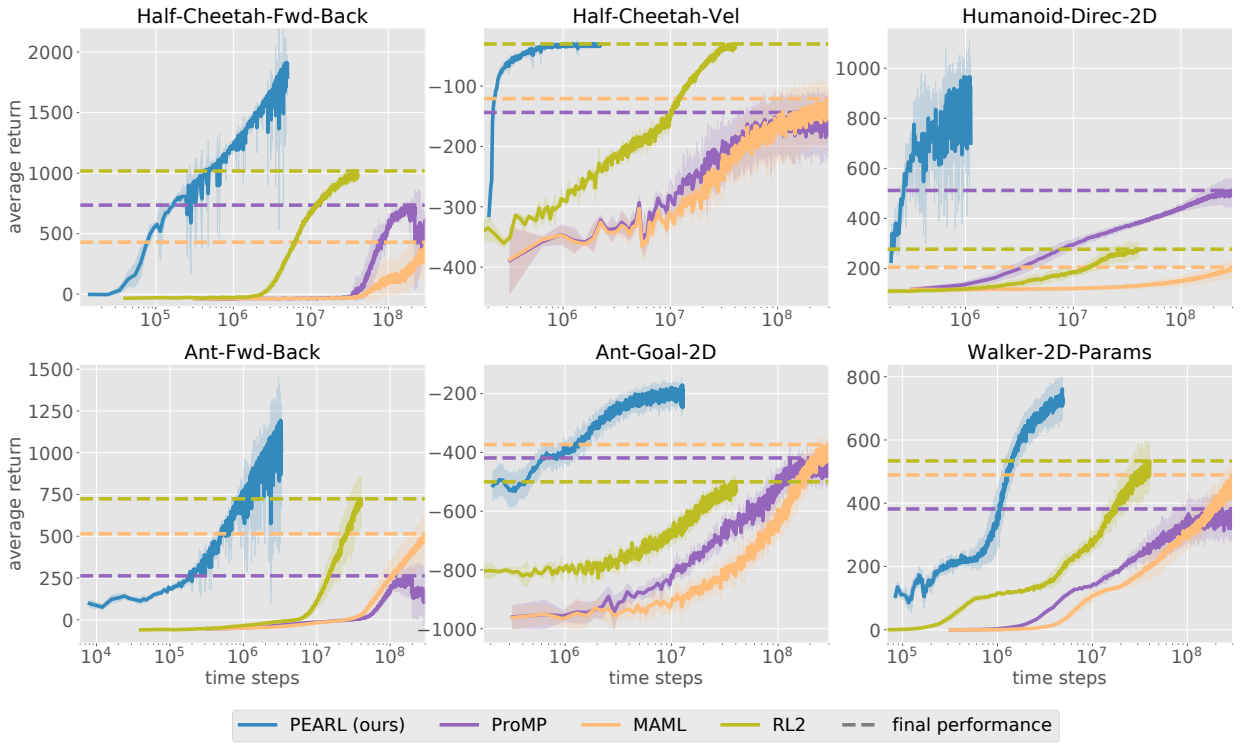


Figure 34: **Meta-learning continuous control**. Test task performance vs. samples collected during *meta-training*. While in the main paper we truncate the x-axis to better illustrate the performance of PEARL, here we plot PEARL against the on-policy methods run for the full number of time steps (1e8). Note that the x-axis is in **log scale**. PEARL is 20-100 times more sample efficient.

D

APPENDIX: LATENT STATE MODELS FOR META-RL FROM IMAGES

This appendix accompanies the work presented in Chapter 5, Section 5.3.

D.1 MELD IMPLEMENTATION DETAILS

Here, we expand on Section 5.3.2 to provide further implementation details of our algorithm MELD. Also, see our open-source code: https://github.com/tonyzhaozh/meld.

As discussed in Section 5.3.2, the latent state model is comprised of latent dynamics distributions, posterior inference distributions, and generative distributions of observations and rewards. While most timesteps are processed by the same time-invariant dynamics model $p(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1})$ and posterior inference network $q_\phi(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{x}_t, r_t, \mathbf{a}_{t-1})$, we use separate distributions to model the first time step of a trial: $p(\mathbf{s}_1|\mathbf{s}_0)$ and $q_\phi(\mathbf{s}_1|\mathbf{s}_0, \mathbf{x}_1, r_1)$. Note that in Sections 5.3.3 and 5.3.3, we assume trials contain 2 episodes, while in Section 5.3.3 trials contain 1 episode. Following SLAC, we implement two layers of latent variables (please refer to the SLAC paper (A. Lee et al., 2019), for more details). The network $q_\phi(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{x}_t, r_t, \mathbf{a}_{t-1})$ encodes image observations and the network $p(\mathbf{x}_t|\mathbf{s}_t)$ decodes them for the reconstruction loss. Both of these networks include the same convolutional architecture (the decoder simply the transpose of the encoder) that consists of five convolutional layers. The layers have 32, 64, 128, 256, and 256 filters and the corresponding filter sizes are 5, 3, 3, 3, 4. For environments in which the robot observes two images (such as scene and wrist camera), we concatenate the images and apply rectangular filters. All other model networks are fully connected and consist of 2 hidden layers of 32 units each. We use ReLU activations after each layer.

As discussed, we train the actor and critic using the SAC RL algorithm (Haarnoja et al., 2018) with the belief state as input. The SAC algorithm maximizes discounted returns as well as policy entropy via policy iteration. The critic (Q-function) is trained to minimize

the soft Bellman error, which takes the entropy of the policy into account in the backup. We instantiate the actor and critic as fully connected networks with 2 hidden layers of 256 units each. We follow the implementation of SAC, including the use of 2 Q-networks and the tanh actor output activation (please see the SAC paper (Haarnoja et al., 2018) for more details).

Meta-training alternates between collecting data with the current model and policy, and training the model and actor-critic. Gradients from the actor-critic optimization do not flow into the latent state model. Before beginning this alternating scheme, we first train the latent state model on 60 trajectories of data collected with a random policy. Relevant hyper-parameters for meta-training can be found in Table 1.

Table 1: Meta-training hyper-parameters

Parameter	Value
num. training tasks	30
num. eval tasks	10
actor, critic learning rates	3e-4
model learning rate	1e-4
size of per-task replay buffers $\mathcal{B}_i$	1e5
num. tasks collect data	20
num. rollouts per task	1
num. train steps per epoch	640
num. tasks sample for update	20
model batch size	512
actor, critic batch size	512

D.2 SIMULATION EXPERIMENT DETAILS AND ABLATION STUDY

In this section we provide further details on the simulated experiments in Section 5.3.3. We also perform an ablation study of several design decisions in the MELD algorithm.

D.2.1 *Success Metrics*

We define a success metric for each environment that correlates with qualitatively solving the task: Cheetah-vel: within .2m/s of target velocity, Reacher: within 10cm of goal, Peg: complete insertion with 5cm variation possible inside the site, Shelf: mug within 5cm of goal. We use these success metrics because task reward is often misleading when averaged across a distribution of tasks; in peg-insertion, for example, the numerical difference between always inserting the peg correctly versus never inserting it can be as low as 0.1, since the distance between the center of the goal distribution and each goal is quite small and accuracy is required. In Figure 23, we plot this success threshold as a dashed black line.

D.2.2 *Environment Details*

In the Cheetah-vel environment, we control the robot by commanding the torques on the robot's 6 joints. The reward function consists of the difference between the target velocity v_{target} and the current velocity v_x of the cheetah's center of mass, as well a small control cost on the torques sent to the joints:

$$r_{\text{cheetal-vel}} = -|v_x - v_{\text{target}}| + 0.01\|a_t\|_2. \tag{34}$$

The episode length is 50 time steps, and the observation consists of a single 64x64 pixel image from a tracking camera (as shown in Fig. 35a).

In all three Sawyer environments, we control the robot by commanding joint delta-positions for all 7 joints. The reward function indicates the difference between the current end-effector pose x_{ee} and a goal pose x_{goal}, as follows:

$$r_{\text{sawyer-envs}} = -(d^2 + \log(d + 1\text{e-}5)), \quad \text{where } d = \|x_{ee} - x_{\text{goal}}\|_2. \tag{35}$$

This reward function encourages precision near the goal, which is particularly important for the peg insertion task. We impose a maximum episode length of 40 time steps for these environments. The observations for all three of these environments consist of two images concatenated to form a 64x128 image: one from a fixed scene camera, and one from a wrist-mounted first-person view camera. These image observations for each environment are shown in Fig. 35b-d. The simulation time step and control frequency for each of these simulated environments is listed in Table 2.

Table 2: Simulation Environments

Environment	Sim. time step	Control freq.
Cheetah-vel	0.01	10Hz
Reacher	0.0025	4Hz
Peg-insert	0.0025	4Hz
Shelf-placing	0.0025	4Hz

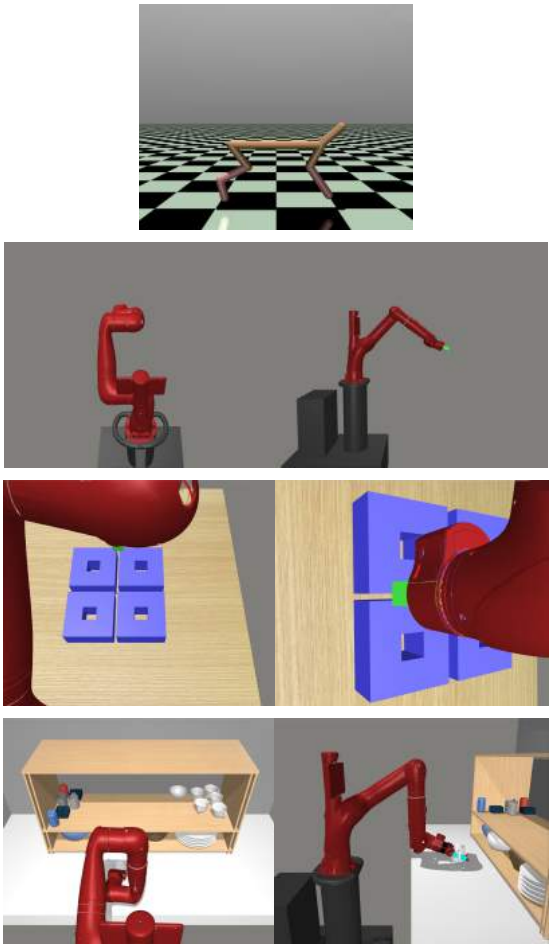


Figure 35: 64x64 and 64x128 image observations, seen as input by MELD for (a) Cheetah-vel, (b) Reacher, (c) Peg-insertion, and (d) Shelf-placing.

D.2.3 *Ablation Study*

In this section we ablate several hyper-parameters of the algorithm to understand which components affect its performance. First, we examine the effect of the number of meta-training tasks on MELD's performance on test tasks. In Figure 36 (left) we plot the performance on test tasks for different numbers of train tasks on the Reacher problem. We find that while generalization fails completely with only a single meta-training task, there are diminishing returns for increasing the number of training tasks beyond 20. This result demonstrates promising generalization, since a relatively low number of training tasks is actually needed in order to be able to solve the held-out test tasks. This result also has a practical benefit in that it precludes the need for instrumenting a large task distribution for meta-training in the real world, which can be cumbersome.

Although the reference implementation of SAC calls for each data collection step to be interleaved with a training step, this process of stopping after each environment step in order to perform training is not possible in the real world. Instead, in MELD, we collect batches of data that consist of full episodes, and we interleave these collection phases with training phases that consist of many gradient steps. Here, we examine the affect of this choice on the performance of the algorithm in simulation. From the results shown in Figure 36 (middle), we see that MELD is not too sensitive to scaling between this type of batched off-policy training and the original SAC-style on-policy training, which is essential for training in the real world.

In Figure 36 (right), we examine the effect of the dimension of the latent variable $\mathbf{s}_t$. These experiments show that a larger latent dimension performs better; we use dimension 256 in all our experiments.

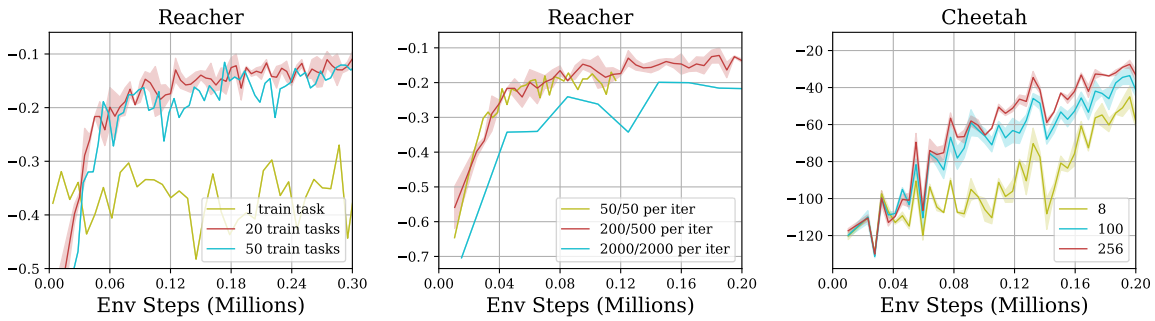


Figure 36: **(left) Number of meta-training tasks** Generalization to evaluation tasks requires training across set of tasks, but increasing beyond 20 tasks does not improve. **(middle) Data / gradients ratio** MELD is not very sensitive to scaling the batch size of data collected, along with the amount of actor-critic training, at each iteration. **(right) Latent variable dimension** Latent state dimension affects task performance, larger is better.

D.3 SPARSE REWARD METHOD AND EXPERIMENT DETAILS

As described in Section 5.3.3, when reward is given only upon completion of the task, efficient exploration is required to identify a new task within a few trials. The agent can acquire these exploration strategies during meta-training by learning strategies tailored to the task distribution. For example, in the button-pressing problem presented in Section 5.3.3, a learned exploration strategy might try pushing each button in succession, but would not try to e.g., pick up the control panel.

While in principle these behaviors can be acquired by MELD as described in Section 5.3.2, in practice performing RL with sparse rewards at meta-training time presents a significant exploration challenge. In effect, to learn useful exploration strategies for meta-test time, the agent must first explore effectively during meta-training. Because RL with sparse rewards is very inefficient, we follow prior work (Gupta et al., 2018b; Rakelly et al., 2019) and assume access to a shaped reward function during meta-training to help learn these strategies. This setup corresponds to a setting where meta-training is performed in a laboratory with access to instrumentation to calculate the shaped reward, while meta-testing occurs outside the lab where only sparse rewards are available.

To make use of the shaped reward during meta-training time, we follow a two-stage procedure. First, we perform meta-training using the shaped reward as prescribed in Section 5.3.2. We then add data collected by this agent to the replay buffer of a second agent, which is trained with a small modification made to the model training loss from Equation 15. Here, the latent state model takes the **sparse** reward signal as input (to

match our desired meta-testing setup), but we still use the shaped reward for the reconstruction target. We denote the shaped reward as $\tilde{r}$ and the sparse reward as r, and highlight the difference from Equation 15 in blue.

$$\mathcal{L}_{model}(\mathbf{x}_{1:T}, r_{1:T}, \tilde{r}_{1:T}, \mathbf{a}_{1:T-1}) = \mathop{\mathbb{E}}_{\mathbf{s}_{1:T}\sim q_\phi} \sum_{t=1}^{T} \log p_\phi(\mathbf{x}_t|\mathbf{s}_t) + \log p_\phi(\tilde{r}_t|\mathbf{s}_t)$$
$$- D_{KL}(q_\phi(\mathbf{s}_1|\mathbf{x}_1, r_1)\|p(\mathbf{s}_1)) - \sum_{t=2}^{T} D_{KL}(q_\phi(\mathbf{s}_t|\mathbf{x}_t, r_t, \mathbf{s}_{t-1}, \mathbf{a}_{t-1})\|p_\phi(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1})). \quad (36)$$

Additionally, we use the shaped reward to train the critic, since this is also not required at meta-test time.

Note that MELD does *not* simply learn to copy the trajectories from the shaped-reward training that were used to warmstart the sparse-reward training. The former ("expert") trajectories move from the starting position directly to the correct button, since the shaped reward contains information to almost immediately identify the correct task. The MELD trajectories that result from receiving only sparse rewards as input, however, demonstrate systematic exploration of visiting different buttons in order to determine the correct one (see Figure 24). Finally, note that we use this same approach for the real-world WidowX experiments in Section 5.3.3.

D.4 Importance of performing inference at every timestep

In this section, we discuss the benefit of the time-varying nature of MELD's latent belief. We emphasize that this design decision is useful in the standard meta-RL setting, as well as in other realistic settings of the underlying task itself changing within an episode.

D.4.1 *Fast Adaptation*

We first present a didactic image-based 2D navigation problem to illustrate how MELD can learn extended exploration strategies to adapt to a new task, similar to the results shown in Figure 24. Here, the task distribution consists of goals located along a semi-circle around the start state. The agent receives inputs in the form of 64x64 image observations and rewards that are non-zero only upon reaching the correct goal. We use the same approach to using shaped reward for meta-training as described in Appendix D.3.

As shown in Figure 37a, the agent learns an efficient exploration strategy of traversing the semi-circle goal region until the goal is found. By updating the posterior belief at each step, MELD is able to find the goal within $10-20$ steps, instead of multiple episodes as required by methods that explore via posterior sampling (Rakelly et al., 2019; Gupta et al., 2018b) that hold the task variable constant across each episode. Furthermore, note that once the goal is found, MELD can navigate directly to it (Figure 37b) in the next episode.

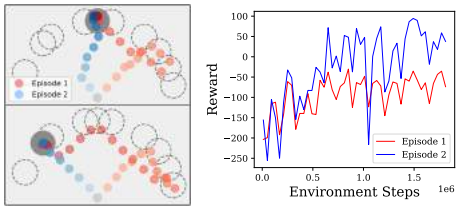


Figure 37: Image-based 2D navigation with reward given after finding the right goal: (Left) trajectory traces of meta-learned exploration finding goal in Episode-1 (red) and going directly there in Episode-2 (blue). (Right) higher rewards in episode-2 show preservation of information across episodes.

This experiment demonstrates that even in standard meta-RL setting where the underlying task remains constant throughout an episode, updating task information at each timestep can enable faster adaptation. We argue that this behavior has safety benefits in the real world, since the agent need not complete full episodes of potentially hazardous exploration before incorporating task information.

D.4.2 *Adapting to Task Changes within Episodes*

The experiments in the main paper as well as the section above consider the standard meta-RL paradigm, where the agent adapts to one test task at a time. However, many realistic scenarios consist of a sequence of tasks. For example, consider a robot moving a mug filled with liquid; if some of the liquid spills, the robot must adapt to the new dynamics of the lighter mug to finish the job. Because MELD updates the belief over the hidden variables at each time step, it can be directly applied to this setting without modification. We evaluate MELD in the Cheetah-vel environment

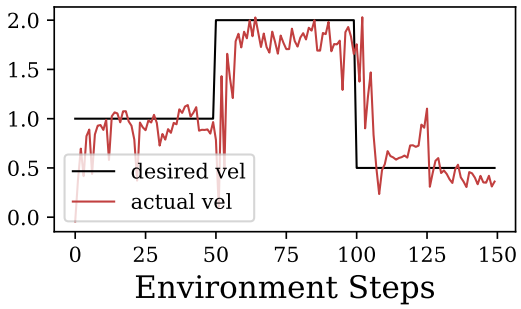


Figure 38: MELD tracking changing velocity targets for Cheetah-vel.

on a sequence of 3 different target velocities within a single episode and observe in Figure 38 that MELD adapts to track each velocity within a few time steps.

D.5 TASK RESET MECHANISM FOR WIDOWX EXPERIMENTS

Since meta-training requires training across a distribution of tasks, we build an automatic task reset mechanism for the real-world experiments with the WidowX robot performing ethernet cable insertion. This mechanism controls the translational and rotational displacement of the network switch. The network switch(A) is mounted to a 3D printed housing(B) with gear attached. We control the rotation of the housing through motor 1. This setup is then mounted on top of a linear rail(C) and motor 2 controls its translational displacement through a timing pulley. In our experiments, the training task distribution consisted of 20 different tasks, where each task was randomly assigned from a rotational range of 16 degrees and a translational range of 2cm.

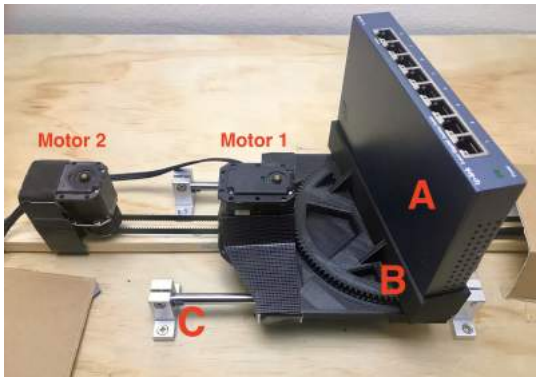


Figure 39: **Automatic task reset mechanism** The network switch is rotated and translated by a series of motors in order to generate different tasks for meta-learning. This allows our meta-learning process to be entirely automated, without needing human intervention to reset either the robot or the task at the beginning of each rollout.

D.6 SAWYER MULTI-TASK PEG INSERTION

In this experiment, we demonstrate that MELD can reason jointly about state and task information to perform real-world peg insertion with a 7-DoF Sawyer robot (Figure 40, left). On the Sawyer robot, we learn precise peg insertion where the task distribution consists of three tasks, each corresponding to a different target box. Note that the goal is not provided to the agent, but must be inferred from its history of observations and rewards.

The reward function is the sum of the L2-norms of translational and rotational distances between the pose of the object in the end-effector and a goal pose. The agent's observations are concatenated images from two webcams (Figure 40, right): one fixed view and one first-person view from a wrist-mounted camera. The robot succeeds on all three tasks after training on 4 hours of data (60,000 samples at 4Hz), as shown in Figure 41. Videos of the experiment may be found on the project website.

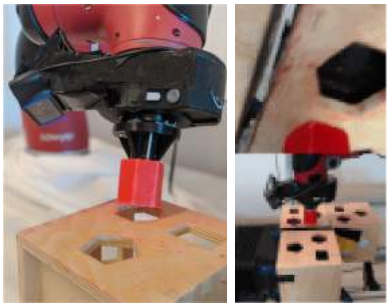


Figure 40: **(left)** Performing precise peg insertion in the real world with a 7-DoF Sawyer robot. **(right)** 64x128 image observations seen as input by the Sawyer.

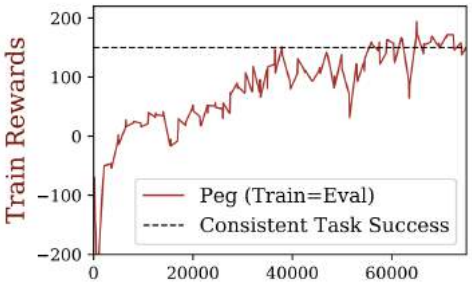


Figure 41: Rewards on train tasks during meta-training for Sawyer peg insertion.