Learning a Prior over Intent
via Meta-Inverse Reinforcement Learning

Kelvin Xu [1] **Ellis Ratner** [1] **Anca Dragan** [1] **Sergey Levine** [1] **Chelsea Finn** [1]

Abstract

A significant challenge for the practical application of reinforcement learning to real world problems is the need to specify an oracle reward function that correctly defines a task. Inverse reinforcement learning (IRL) seeks to avoid this challenge by instead inferring a reward function from expert demonstrations. While appealing, it can be impractically expensive to collect datasets of demonstrations that cover the variation common in the real world (e.g. opening any type of door). Thus in practice, IRL must commonly be performed with only a limited set of demonstrations where it can be exceedingly difficult to unambiguously recover a reward function. In this work, we exploit the insight that demonstrations from other tasks can be used to constrain the set of possible reward functions by learning a "prior" that is specifically optimized for the ability to infer expressive reward functions from limited numbers of demonstrations. We demonstrate that our method can efficiently recover rewards from images for novel tasks and provide intuition as to how our approach is analogous to learning a prior.

1. Introduction

Reinforcement learning (RL) algorithms have the potential to automate a wide range of decision-making and control tasks across a variety of different domains, as demonstrated by successful recent applications ranging from robotic control (Kober & Peters, 2012; Levine et al., 2016) to game playing (Mnih et al., 2015; Silver et al., 2016). A key assumption of the RL problem statement is the availability of a reward function that accurately describes the desired task. For many real world tasks, reward functions can be chal-
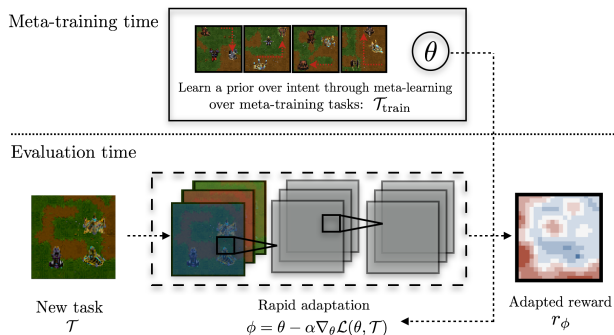

Figure 1. A diagram of our meta-inverse RL approach. Our approach attempts to remedy over-fitting in few-shot IRL by learning a "prior" that constraints the set of possible reward functions to lie within a few steps of gradient descent. Standard IRL attempts to recover the reward function directly from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

lenging to manually specify, while being crucial for good performance (Amodei et al., 2016). Most real world tasks are multifaceted and require reasoning over multiple factors in a task (e.g. a robot cleaning in a house with children), while simultaneously providing appropriate reward shaping to make the task feasible with tractable exploration (Ng et al., 1999). These challenges are compounded by the inherent difficulty of specifying rewards for tasks with high-dimensional observation spaces such as images.

Inverse reinforcement learning (IRL) is an approach that aims to address this problem by instead inferring the reward function from demonstrations of the task (Ng & Russell, 2000). This has the appealing benefit of taking a data-driven approach to reward specification in place of hand engineering. In practice however, rewards functions are rarely learned as it can be prohibitively expensive to provide demonstrations that cover the variability common in real world tasks (e.g., collecting demonstrations of opening every type of door knob). In addition, while learning a complex function from high dimensional observations might make an expressive function approximator seem like a reasonable modelling assumption, in the "few-shot" domain it is notoriously difficult to unambiguously recover a good reward

[1]Department of Electrical Engineering and Computer Science, University of California, Berkeley, USA. Correspondence to: Kelvin Xu <kelvinxu@berkeley.edu>.

function with expressive function approximators. Many prior approaches have thus relied on low-dimensional linear models with handcrafted features that effectively encode a strong prior on the relevant features of a task. This requires engineering a set of features by hand that work well for a specific problem. In this work, we propose an approach that instead explicitly learns expressive features that are robust even when learning with limited demonstrations.

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

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

2. Related Work

Inverse reinforcement learning (IRL) (Ng & Russell, 2000) is the problem of inferring an expert's reward function directly from demonstrations. Prior methods for performing IRL range from margin based approaches (Abbeel & Ng, 2004; Ratliff et al., 2006) to probabilistic approaches (Ramachandran & Amir, 2007; Ziebart et al., 2008). Although it is possible to extend our approach to any other IRL method, in this work we base on work on the maximum entropy (MaxEnt) framework (Ziebart et al., 2008). In addition to allowing for sub-optimality in the expert demonstrations, MaxEnt-IRL can be re-framed as a maximum likelihood estimation problem. (Sec. 3).

In part to combat the under-specified nature of IRL, prior work has often used low-dimensional linear parameterizations with handcrafted features (Abbeel & Ng, 2004; Ziebart et al., 2008). In order to learn from high dimensional input, Wulfmeier et al. (2015) proposed applying fully convolutional networks (Shelhamer et al., 2017) to the MaxEnt IRL framework (Ziebart et al., 2008) for several navigation tasks (Wulfmeier et al., 2016b;a). Other methods that have incorporated neural network rewards include guided cost learning (GCL) (Finn et al., 2017a), which uses importance sampling and regularization for scalability to high-dimensional spaces, and adversarial IRL (Fu et al., 2018). Several other methods have also proposed imitation learning approaches based on adversarial frameworks that resemble IRL, but do not aim to directly recover a reward function (Ho & Ermon, 2016; Li et al., 2017; Hausman et al., 2017; Kuefler & Kochenderfer, 2018). In this work, instead of improving the ability to learn reward functions on a single task, we focus on the problem of effectively learning to use prior demonstration data from other IRL tasks, allowing us to learn new tasks from a limited number demonstrations even with expressive non-linear reward functions.

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

Our approach builds on work on the broader problem of meta-learning (Schmidhuber, 1987; Bengio et al.; Naik & Mammone, 1992; Thrun & Pratt, 2012) and generative modelling (Rezende et al., 2016; Reed et al., 2018; Mordatch, 2018). Prior work has proposed a variety of solutions for learning to learn including memory based methods (Duan et al., 2016; Santoro et al., 2016; Wang et al., 2016; Mishra et al., 2017), methods that learn an optimizer and/or initialization (Andrychowicz et al., 2016; Ravi & Larochelle, 2016; Finn et al., 2017a; Li & Malik, 2017), and methods that compare new datapoints in a learned metric space (Koch, 2015; Wang & Hebert, 2016; Vinyals et al., 2016;

Shyam et al., 2017; Snell et al., 2017). Our work is motivated by the goal of broadening the applicability of IRL, but in principle it is possible to adapt many of these meta-learning approaches for our problem statement. We build upon Finn et al. (2017a), which has also been previously applied to the related problems of imitation learning and human motion prediction (Wang et al., 2016; Finn et al., 2017b; Alet et al., 2018). We leave it to future work to do a comprehensive investigation of different meta-learning approaches which could broaden the applicability of IRL.

3. Preliminaries and Overview

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

3.1. Learning Rewards via Maximum Entropy Inverse Reinforcement Learning

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

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

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

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

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

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

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

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

Learning in general energy-based models of this form is common in many applications such as structured prediction. However, in contrast to applications where learning

can be supervised by millions of labels (e.g. semantic segmentation), the learning problem in Eq. 3 must typically be performed with a relatively small number of example demonstrations. In this work, we seek to address this issue in IRL by providing a way to integrate information from prior tasks to constrain the optimization in Eq. 3 in the regime of limited demonstrations.

3.2. Meta-Learning

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

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

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

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

where the optimization is over an initial set of parameters $\boldsymbol{\theta}$ and the loss on the held out tasks $X_{\mathcal{T}}^{test}$ becomes the signal

for learning the initial parameters for gradient descent (with step size α) on $X_{\mathcal{T}}^{tr}$. This optimization is analogous to adding a constraint in a multi-task setting, which we show in later sections is analogous in our setting to learning a prior over reward functions.

4. Learning to Learn Rewards

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

During meta-training, we have a set of tasks $\{\mathcal{T}_i \; ; \; i = 1..N\}$. Each task $\mathcal{T}_i$ has a set of demonstrations $\mathcal{D}_{\mathcal{T}} = \{\tau_1, \ldots, \tau_K\}$ from an expert policy which we partition into disjoint $\mathcal{D}_{\mathcal{T}}^{tr}$ and $\mathcal{D}_{\mathcal{T}}^{test}$ sets. The demonstrations for each meta-training task are assumed to be produced by the expert according to the maximum entropy model in Section 3.1. During meta-training, these tasks will be used to encodes common structure so that our model can quickly acquire rewards for new tasks from just a few demonstrations.

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

4.1. Meta Reward and Intention Learning (MandRIL)

In order to meta-learn a reward function that can act as a prior for new tasks and new environments, we first formalize the notion of a good reward by defining a loss $\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta})$ on the reward function r_θ for a particular task $\mathcal{T}$. We use the MaxEnt IRL loss $\mathcal{L}_{\text{IRL}}$ discussed in Section 3, which, for a given $\mathcal{D}_{\mathcal{T}}$, leads to the following gradient (Ziebart et al., 2008):

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

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

If our end goal were to achieve a single reward function that

---

Algorithm 1 Meta Reward and Intention Learning (MandRIL)

1: **Input:** Set of meta-training tasks $\{\mathcal{T}\}^{\text{meta-train}}$
2: **Input:** hyperparameters α, β
3: **function** MAXENTIRL-GRAD($r_{\boldsymbol{\theta}}, \mathcal{T}, \mathcal{D}$) $\triangleright$ Single task update
4:     # *Compute state visitations of demos*
5:     $\boldsymbol{\mu}_{\mathcal{D}} = $ STATE-VISITATIONS-TRAJ($\mathcal{T}, \mathcal{D}$)
6:     # *Compute Max-Ent state visitations*
7:     $\mathbb{E}_\tau[\boldsymbol{\mu}_\tau] = $ STATE-VISITATIONS-POLICY($r_{\boldsymbol{\theta}}, \mathcal{T}$)
8:     # *MaxEntIRL gradient* (Ziebart et al., 2008)
9:     $\frac{\partial \mathcal{L}}{\partial r_{\boldsymbol{\theta}}} = \mathbb{E}_\tau[\boldsymbol{\mu}_\tau] - \boldsymbol{\mu}_{\mathcal{D}}$
10:    **Return** $\frac{\partial \mathcal{L}}{\partial r_{\boldsymbol{\theta}}}$

11:
12: Randomly initialize $\boldsymbol{\theta}$
13: **while** not done **do**
14:     Sample batch of tasks $\mathcal{T}_i \sim \{\mathcal{T}\}^{\text{meta-train}}$
15:     **for all** $\mathcal{T}_i$ **do**
16:         Sample demos $\mathcal{D}^{\text{tr}} = \{\tau_1, \ldots, \tau_K\} \sim \mathcal{T}_i$
17:         # *Inner loss computation*
18:         $\frac{\partial \mathcal{L}_{\mathcal{T}_i}^{\text{tr}}(\boldsymbol{\theta})}{\partial r_{\boldsymbol{\theta}}} = $ MAXENTIRL-GRAD($r_{\boldsymbol{\theta}}, \mathcal{T}_i, \mathcal{D}^{\text{tr}}$)
19:         Compute $\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}^{\text{tr}}(\boldsymbol{\theta})$ from $\frac{\partial \mathcal{L}_{\mathcal{T}_i}^{\text{tr}}(\boldsymbol{\theta})}{\partial r_{\boldsymbol{\theta}}}$
20:         Compute $\phi_{\mathcal{T}_i} = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}^{\text{tr}}(\boldsymbol{\theta})$
21:         Sample demos $\mathcal{D}^{\text{test}} = \{\tau_1', \ldots, \tau_{K'}'\} \sim \mathcal{T}_i$
22:         # *Outer loss computation*
23:         $\frac{\partial \mathcal{L}_{\mathcal{T}_i}^{\text{test}}}{\partial r_{\boldsymbol{\theta}}} = $ MAXENTIRL-GRAD($r_{\phi_{\mathcal{T}_i}}, \mathcal{T}_i, \mathcal{D}^{\text{test}}$))
24:         # *Compute meta-gradient*
25:         Compute $\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}^{\text{test}}$ from $\frac{\partial \mathcal{L}_{\mathcal{T}_i}^{\text{test}}}{\partial r_{\boldsymbol{\theta}}}$ via chain rule
26:     Compute update to $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \sum_i \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}^{\text{test}}$

---

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

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

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

lead to a reward function $r_{\phi_{\mathcal{T}}}$ for task $\mathcal{T}$, such that the IRL loss (corresponding to negative log-likelihood) for a disjoint

set of test demonstrations, given by $\mathcal{L}_{\text{IRL}}^{\mathcal{T},\text{test}}$, is minimized. The corresponding optimization problem for $\boldsymbol{\theta}$ can therefore be written as follows:

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

Our method acquires this prior $\boldsymbol{\theta}$ over rewards in the task distribution $p(\mathcal{T})$ by optimizing this loss. This amounts to an extension of the MAML algorithm in Section 3.2 to the inverse reinforcement learning setting. This extension is quite challenging, because computing the MaxEnt IRL gradient requires repeatedly solving for the current maximum entropy policy and visitation frequencies, and the MAML objective requires computing derivatives *through* this gradient step. Next, we describe in detail how this is done. An overview of our method is also outlined in Alg. 1.

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

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

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}}^{\text{test}}(\boldsymbol{\theta}) = \frac{\partial}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}}^{\text{tr}}(\boldsymbol{\theta})) \frac{\partial r_{\boldsymbol{\phi}_{\mathcal{T}}}}{\partial \boldsymbol{\phi}_{\mathcal{T}}} \frac{\partial \mathcal{L}_{\mathcal{T}}^{\text{test}}}{\partial r_{\boldsymbol{\phi}_{\mathcal{T}}}}$$

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

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

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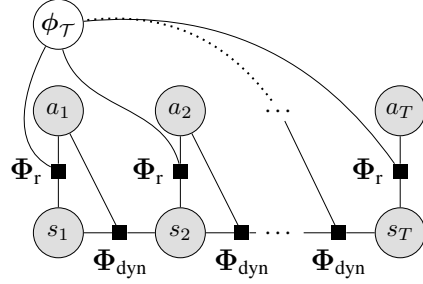

Figure 2. Our approach can be understood as approximately learning a distribution over the demonstrations τ, in the factor graph $p(\tau) = \frac{1}{Z} \prod_{t=1}^{T} \boldsymbol{\Phi}_{\text{r}}(\boldsymbol{\phi}_{\mathcal{T}}, \mathbf{s}_t, \mathbf{a}_t) \boldsymbol{\Phi}_{\text{dyn}}(\mathbf{s}_{t+1},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

A detailed derivation of this expression is provided in the supplementary Appendix E.

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

4.2. Connection to Learning a Prior over Intent

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

By viewing IRL as maximum likelihood estimation in a particular graphical model (Fig. 2), we can take the perspective of Grant et al. (2018) who showed that for a linear model, fast adaptation via a few steps of gradient descent in MAML is performing MAP inference over ϕ, under a Gaussian prior with the mean $\boldsymbol{\theta}$ and a covariance that depends on the step size, number of steps and hessian of the loss. This is based on the connection between early stopping and regularization previously discussed in Santos (1996), which we refer the readers to for a more detailed discussion. The interpretation of MAML as imposing a Gaussian prior on the parameters is exact in the case of a likelihood that is quadratic in the parameters (such as the log-likelihood of a Gaussian in terms of its mean). For any non-quadratic likelihood, this is an approximation in a local neighborhood

around θ (i.e. up to convex quadratic approximation). In the case of complex parameterizations, such as deep function approximators, this is a coarse approximation and unlikely to be the mode of a posterior. However, we can still frame the effect of early stopping and initialization as serving as a prior in a similar way as prior work (Sjöberg & Ljung, 1995; Duvenaud et al., 2016; Grant et al., 2018). More importantly, this interpretation hints at future extensions to our approach that could benefit from employing more fully Bayesian approaches to reward and goal inference.

5. Experiments

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

We test our core hypothesis by comparing learning performance on a new tasks starting from the initialization produced by MandRIL with learning a separate model for every task starting either from a random initialization or from an initialization obtained by supervised pre-training. We refer to these approaches as learning "FROM SCRATCH" and "AVERAGE GRADIENT" pretraining respectively. Our supervised pre-training baseline follows the average gradient during meta-training tasks and finetunes at meta-test time (as discussed in Section 4). Unlike our method, supervised pre-training does not optimize for a model that performs well under fine tuning, but does use the same prior data to pre-train. We additionally compare to pre-training on a single task as well as all the meta-training tasks.

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

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

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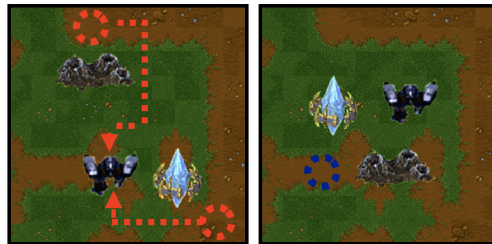

Figure 3. An example task on the SpriteWorld domain. When learning a task, the agent has access to the image (left) and demonstrations (red arrows). To evaluate learning (right), the agent is tested for its ability to recover the reward for the task when the objects have been rearranged. The reward structure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

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

(1) SpriteWorld navigation domain. Since most prior IRL work (and multi-task IRL work) studied settings where linear reward function approximators suffice (i.e., low-dimensional state spaces and hand-designed features), we design an experiment that is significantly more challenging—that requires learning rewards on raw pixels. We consider a navigation problem where we must learn a convolutional neural network that directly maps image pixels to rewards. We introduce a family of tasks called "SpriteWorld." Some example tasks are shown in Fig. 3. Tasks involve navigating to goal objects while exhibiting preference over terrain types (e.g., the agent prefers to traverse dirt tiles over traversing grass tiles). At meta-test time, we provide one or a few demonstrations in a single training environment and evaluate the reward learned using these demonstrations in a new, test environment that contains the same objects as the training environment, but arranged differently. Evaluating in a new test environment is critical to measure that, after adapting to the training environment from a few demonstrations, the reward learned the correct visual cues, rather than simply memorizing the demonstration trajectory.

We generate unique tasks in this domain as follows. First, we randomly choose a set of three sprites from one hundred sprites from the original game (creating a total of 161,700 unique tasks). We randomly place these three sprites within a randomly generated terrain tiling; we designate one of the sprites to be the goal landmark of the navigation task. The other two objects are treated as obstacles for which the agent incurs a large negative reward for not avoiding. In each task, we optimize our model on a training world and generalization in a test world, as described below.
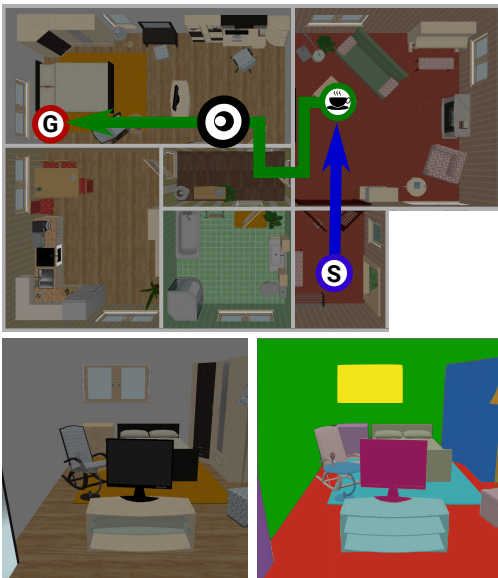
Figure 4. An example task in the SUNCG environment (top). The agent must complete either a "NAV" task (blue line) where the goal is to navigate from the start to the cup or a "PICK" task (blue + green line) where the agent must also bring the cup to the bed. The agent's observation is a panoramic first-person viewpoint (see bottom left for RGB). Following the convention in prior work (Fu et al., 2019), we provide to the reward function the corresponding semantic images (bottom right). These images are 32×24 containing 61 channels corresponding to each object class.

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

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

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

Evaluation protocol: We evaluate on held-out tasks that were unseen during meta-training. In the SpriteWorld domain, we consider two settings: (1) tasks involving new combinations and placements of sprites, with sprites that were present during meta-training, and (2) tasks with combinations of unseen sprites which we refer to as "out of domain objects." For each task, we generate one environment (a set
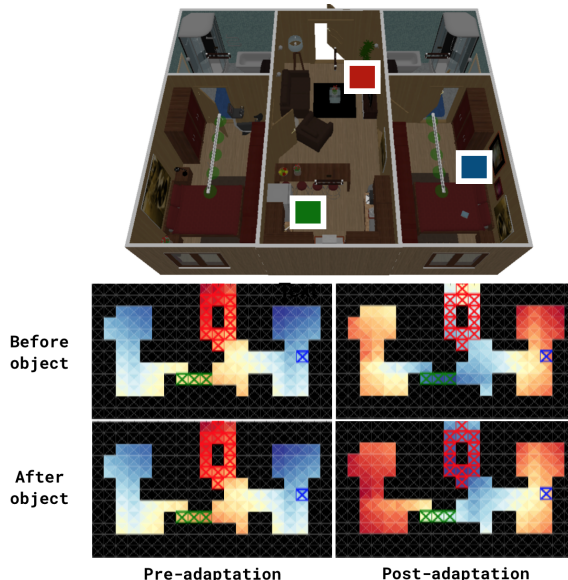

Figure 5. An example adaptation in an UNSEEN-HOUSE (best viewed in color). The agent starts in one room (blue square) and is required to pick up the vase (green square) and take it to the living room (red square). The value function (blue is high, red is low) under the learned reward (bottom) exhibits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

of sprite positions) along with demonstrations for adapting the reward, and generate a second environment (with new sprite positions) for evaluating the adapted reward.

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

Evaluation Metrics. We measure performance using the expected value difference, which measures the suboptimality of a policy learned under the learned reward; this is a performance metric used in prior IRL work (Levine et al., 2011; Wulfmeier et al., 2015). The metric is computed by taking the difference between the value of the optimal policy under the learned reward and the value of the optimal policy under the true reward. On the SUNCG domain, we follow Fu et al. (2019) and report the success rate of the optimal policy under the learned reward function.
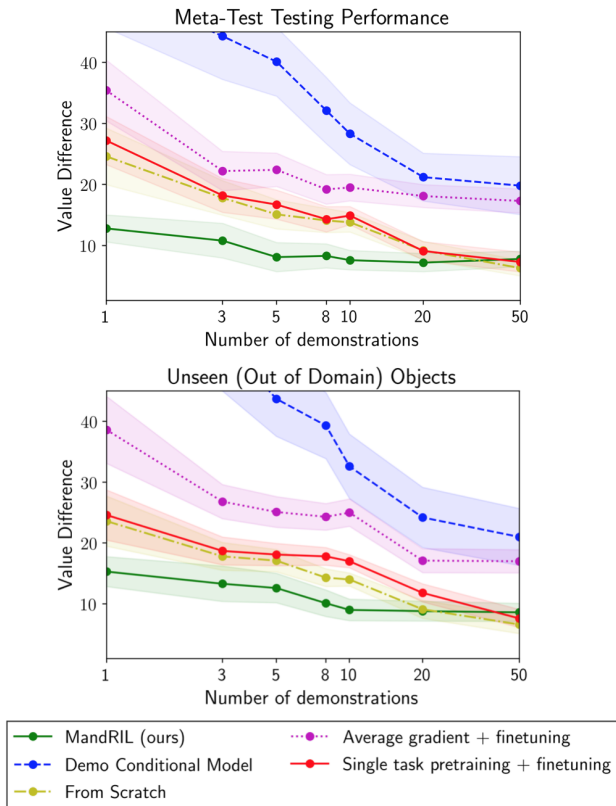
Figure 6. Meta-test performance on the SpriteWorld domain (lower is better): held-out tasks performance (top) and held-out tasks with novel sprites (bottom). The recurrent meta-learner has a value difference > 60 in both test settings. In both test settings, MandRIL achieves comparable performance to the training environment, while the other methods overfit until they receive at least 10 demonstrations (see Appendix C for training environment performance). We find that pre-training on the full set of tasks leads to negative transfer, while pre-training on a single task is comparable to random initialization. ManDRIL outperforms both alternative initialization approaches, which shows that optimizing for initial weights for fine-tuning robustly improves performance. Shaded regions show 95% confidence intervals.

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

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

METHOD	TEST			UNSEEN HOUSES		
	PICK	NAV	TOTAL	PICK	NAV	TOTAL
BEHAVIORAL CLONING	0.4	8.2	4.3	3.7	12.0	9.4
MAXENT IRL (AVG GRADIENT)	37.3	83.7	60.8	38.3	89.7	73.3
MAXENT IRL (FROM SCRATCH)	42.4	87.9	65.4	48.1	89.9	76.5
MANDRIL(OURS)	**52.3**	**90.7**	**77.3**	**56.3**	**91.0**	**82.6**
MANDRIL (PRE-ADAPTATION)	6.0	35.3	20.7	4.3	34.6	25.3

of MandRIL on the out of distribution test setting (Fig. 6, bottom): although the evaluation is on new sprites, MandRIL is still able to adapt via gradient descent and exceed the performance all other methods.

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

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

6. Conclusion

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

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML, New York, NY, USA, 2004.

Alet, F., Lozano-Pérez, T., and Kaelbling, L. P. Modular meta-learning. *arXiv preprint arXiv:1806.10166*, 2018.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., and de Freitas, N. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pp. 3981–3989, 2016.

Babeş-Vroman, M., Marivate, V., Subramanian, K., and Littman, M. Apprenticeship learning about multiple intentions. In *International Conference on International Conference on Machine Learning*, ICML, USA, 2011.

Baker, C., B Tenenbaum, J., and R Saxe, R. Goal inference as inverse planning. 01 2007.

Bengio, Y., Bengio, S., and Cloutier, J. *Learning a synaptic learning rule*.

Choi, J. and Kim, K.-E. Nonparametric bayesian inverse reinforcement learning for multiple reward functions. In *Advances in Neural Information Processing Systems*, NIPS, USA, 2012.

Dimitrakakis, C. and Rothkopf, C. A. Bayesian multitask inverse reinforcement learning. In *European Conference on Recent Advances in Reinforcement Learning*, EWRL, 2012.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Duan, Y., Andrychowicz, M., Stadie, B. C., Ho, J., Schneider, J., Sutskever, I., Abbeel, P., and Zaremba, W. One-shot imitation learning. In *NIPS*, pp. 1087–1098, 2017.

Duvenaud, D., Maclaurin, D., and Adams, R. Early stopping as nonparametric variational inference. In *Artificial Intelligence and Statistics*, pp. 1070–1077, 2016.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017a.

Finn, C., Yu, T., Zhang, T., Abbeel, P., and Levine, S. One-shot visual imitation learning via meta-learning. *arXiv preprint arXiv:1709.04905*, 2017b.

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adversarial inverse reinforcement learning. *International Conference on Learning Representations*, 2018.

Fu, J., Korattikara, A., Levine, S., and Guadarrama, S. From language to goals: Inverse reinforcement learning for vision-based instruction following. *International Conference on Learning Representations (ICLR)*, 2019.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Grant, E., Finn, C., Levine, S., Darrell, T., and Griffiths, T. Recasting gradient-based meta-learning as hierarchical bayes. *International Conference on Learning Representations (ICLR)*, 2018.

Hausman, K., Chebotar, Y., Schaal, S., Sukhatme, G., and Lim, J. J. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 1235–1245, 2017.

Ho, J. and Ermon, S. Generative adversarial imitation learning. In *Neural Information Processing Systems (NIPS)*, 2016.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kober, J. and Peters, J. Reinforcement learning in robotics: A survey. In *Reinforcement Learning*, pp. 579–610. Springer, 2012.

Koch, G. Siamese neural networks for one-shot image recognition. 2015.

Kuefler, A. and Kochenderfer, M. J. Burn-in demonstrations for multi-modal imitation learning. 2018.

Levine, S., Popovic, Z., and Koltun, V. Nonlinear inverse reinforcement learning with gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 19–27, 2011.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research (JMLR)*, 17(39):1–40, 2016.

Li, K. and Burdick, J. W. Meta inverse reinforcement learning via maximum reward sharing for human motion analysis. *CoRR*, abs/1710.03592, 2017.

Li, K. and Malik, J. Learning to optimize neural nets. *arXiv preprint arXiv:1703.00441*, 2017.

Li, Y., Song, J., and Ermon, S. Inferring the latent structure of human decision-making from raw visual inputs. *arXiv preprint arXiv:1703.08840*, 2017.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. Meta-learning with temporal convolutions. *arXiv preprint arXiv:1707.03141*, 2017.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Mordatch, I. Concept learning with energy-based models. In *ICLR Workshop*, 2018.

Naik, D. K. and Mammone, R. Meta-neural networks that learn by learning. In *Neural Networks, 1992. IJCNN., International Joint Conference on*, volume 1, pp. 437–442. IEEE, 1992.

Ng, A. Y. and Russell, S. J. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. 1999.

Ramachandran, D. and Amir, E. Bayesian inverse reinforcement learning. In *International Joint Conference on Artifical Intelligence*, San Francisco, CA, USA, 2007.

Ratliff, N. D., Bagnell, J. A., and Zinkevich, M. A. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pp. 729–736. ACM, 2006.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. 2016.

Reed, S., Chen, Y., Paine, T., van den Oord, A., Vinyals, O., Eslami, S. A., Rezende, D., and de Freitas, N. Few-shot autoregressive density estimation: Towards learning to learn distributions. In *ICLR*, 2018.

Rezende, D. J., Mohamed, S., Danihelka, I., Gregor, K., and Wierstra, D. One-shot generalization in deep generative models. *arXiv preprint arXiv:1603.05106*, 2016.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635, 2011.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, 2016.

Santos, R. J. Equivalence of regularization and truncated iteration for general ill-posed problems. *Linear algebra and its applications*, 236:25–33, 1996.

Schmidhuber, J. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

Shelhamer, E., Long, J., and Darrell, T. Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39 (4), April 2017.

Shyam, P., Gupta, S., and Dukkipati, A. Attentive recurrent comparators. *arXiv preprint arXiv:1703.00767*, 2017.

Siegelmann, H. T. and Sontag, E. D. On the computational power of neural nets. *Journal of computer and system sciences*, 50(1):132–150, 1995.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Sjöberg, J. and Ljung, L. Overtraining, regularization and searching for a minimum, with application to neural networks. *International Journal of Control*, 62(6):1391–1407, 1995.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 4080–4090, 2017.

Song, S., Yu, F., Zeng, A., Chang, A. X., Savva, M., and Funkhouser, T. Semantic scene completion from a single depth image. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Synnaeve, G., Nardelli, N., Auvolat, A., Chintala, S., Lacroix, T., Lin, Z., Richoux, F., and Usunier, N. Torchcraft: a library for machine learning research on real-time strategy games. *arXiv preprint arXiv:1611.00625*, 2016.

Thrun, S. and Pratt, L. *Learning to learn*. Springer Science & Business Media, 2012.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pp. 3630–3638, 2016.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Wang, Y. and Hebert, M. Learning to learn: Model regression networks for easy small sample learning. In *European Conference on Computer Vision (ECCV)*, October 2016.

Wulfmeier, M., Ondruska, P., and Posner, I. Maximum entropy deep inverse reinforcement learning. In *Neural Information Processing Systems Conference, Deep Reinforcement Learning Workshop*, volume abs/1507.04888, 2015.

Wulfmeier, M., Rao, D., and Posner, I. Incorporating human domain knowledge into large scale cost function learning. *CoRR*, abs/1612.04318, 2016a.

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

Zaremba, W., Sutskever, I., and Vinyals, O. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

Ziebart, B. D., Maas, A., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08. AAAI Press, 2008.