# Demonstration-Conditioned Reinforcement Learning for Few-Shot Imitation

**Théo Cachet** [1] **Julien Perez** [1] **Christopher R. Dance** [1]

## Abstract

In *few-shot imitation*, an agent is given a few demonstrations of a previously unseen task, and must then successfully perform that task. We propose a novel approach to learning few-shot-imitation agents that we call *demonstration-conditioned reinforcement learning* (DCRL). Given a training set consisting of demonstrations, reward functions and transition distributions for multiple tasks, the idea is to define a policy that takes demonstrations and current state as inputs, and to train this policy to maximize the average of the cumulative reward over the set of training tasks. Compared to concurrent approaches, DCRL has several advantages, such as the ability to improve upon suboptimal demonstrations, to operate given state-only demonstrations, and to cope with a domain shift between the demonstrator and the agent. Moreover, we show that DCRL outperforms methods based on behaviour cloning by a large margin, on navigation tasks and on robotic manipulation tasks from the Meta-World benchmark.

## 1. Introduction

We humans owe our success to our uniquely developed ability to learn from others (Boyd et al., 2011), a core component of which is our capacity to imitate (Hoehl et al., 2019; Charpentier et al., 2020). While humans often only need a few demonstrations to learn to perform a task, state-of-the-art imitation learning methods often require prohibitively many demonstrations even to learn simple tasks (Arora et al., 2020). This has motivated the study of *few-shot imitation*, in which the aim is to maximize the expected performance of an agent that must complete a previously unseen task, having only seen a few demonstrations of that task. For instance, a person might demonstrate how to close a specific window and then expect a robot to close that window, even though the robot has never been trained on this task before, and even though the window's initial state may not be the same as it was in the demonstration.

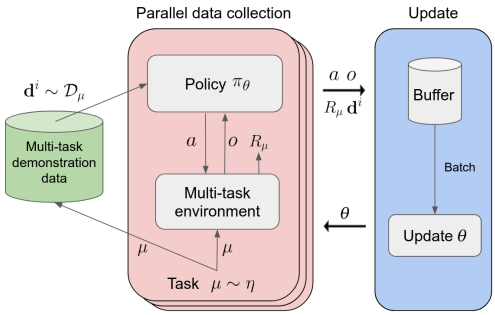


*Figure 1.* The proposed DCRL algorithm, which uses both expert demonstrations and environment interactions to train. The notation is defined in Section 3.

Few-shot imitation has been studied since Duan et al. (2017), motivated by the desire to enable artificial agents to perform a diverse range of tasks, and the hope that a single few-shot imitation agent, trained on a representative set of tasks, can build a representation that enables it to generalize to large numbers of new tasks with acceptable numbers of demonstrations. Existing few-shot imitation algorithms either use *behaviour cloning* (BC), in which the agent is trained to minimize an action-prediction loss (Finn et al., 2017b), or they use *inverse reinforcement learning* (IRL) to infer a reward function from the demonstrations (Goo & Niekum, 2019; Yu et al., 2019a) and then train a policy for that reward function. Unfortunately, as discussed in Section 2, these existing algorithms suffer from one or more of the following limitations. They assume actions are part of the demonstration or that different tasks share a common transition distribution, they ignore domain shift between the agent and the demonstrator, they cannot improve upon suboptimal demonstrators, or they must train a policy each time they are presented with demonstrations of a new task. All these limitations are practically important to address, for instance for robots inferring policies from human demonstrations.

**Proposed Approach.** A key insight of our work is that imitation learning is not a necessary component of few-shot imitation. While the two existing approaches to few-shot imitation (BC and IRL) are also the main approaches to single-

[1]NAVER LABS Europe, 6 chemin de Maupertuis, Meylan, 38240, France. Website:europe.naverlabs.com. Correspondence to: Théo Cachet <theo.cachet@naverlabs.com>.

*Proceedings of the $38^{th}$ International Conference on Machine Learning*, PMLR 139, 2021. 

task imitation learning (Ho & Ermon, 2016), here we propose a third approach. We call this approach *demonstration-conditioned reinforcement learning* (DCRL), by analogy with goal-conditioned reinforcement learning (Nair et al., 2018).

In DCRL, we are given a training set consisting of demonstrations, reward functions and transition distributions, for multiple tasks, as shown in Figure 1. The idea is to work with policies that take demonstrations as input (in addition to the agent's state or observation-action history), and to train such a policy to maximize the average cumulative reward over the set of training tasks. At test time, DCRL requires *neither access to a reward function, nor additional exploration* or interaction with the test environment. Rather, one simply feeds demonstrations of a new task to the trained policy, along with observations of the state, and acts accordingly. In this paper, as we focus on few-shot imitation, demonstrations serve to inform the agent about the task's objective, they serve as examples of how to perform the task, and they may also be informative about the transition distribution if this differs from one task to another. However, DCRL could also be directly applied in situations with a different relationship between demonstrations and tasks, such as avoidance learning (Venuto et al., 2019).

DCRL has several advantages over existing approaches. With no special modifications to the basic algorithm, it accepts demonstrations that consist of state-only observations, it can address situations with a domain shift between the demonstrator and the agent, it can improve upon suboptimal demonstrations, and it requires no additional training when presented with demonstrations of a new task.

As a single demonstration may not be enough to impart the objective of a new task, we propose the use of *cross-demonstration attention* over multiple input demonstrations. While previous work on few-shot imitation has explored policies that use transformers (Dasari & Gupta, 2020; Cachet et al., 2020), the computational cost of such architectures is prohibitive for inputs consisting of multiple demonstrations, each of which is a multivariate time series. Therefore we explore the use of transformers with *axial attention* (Ho et al., 2019), which provide a more efficient alternative architecture for such inputs, as they attend to the temporal and demonstration dimensions of the input independently. The benefit of this approach is particularly clear in our navigation experiments.

**Contributions.** We make the following contributions.

- We propose DCRL, a new approach to learning agents for few-shot imitation.
- We explore the use of cross-demonstration attention, enabled by transformers with axial attention, which have not previously been considered as components of policy architectures.
- We present results on robotic manipulation and navigation benchmarks, demonstrating DCRL's superior performance compared with state-of-the-art alternatives, as well as its ability to improve on suboptimal demonstrations and to cope with domain shifts.

**Organization.** We relate DCRL to previous work (Section 2) and present the proposed method (Section 3). Then we discuss our experiments (Section 4 and Supplementary Material), and conclude (Section 5).

## 2. Related Work

Imitation learning has been studied for decades (Pomerleau, 1991; Abbeel & Ng, 2004; Ross & Bagnell, 2010), and remains an area of active fundamental research (Rajaraman et al., 2020). The following discussion focuses primarily on few-shot imitation, which has been studied since Duan et al. (2017). We also discuss the relationship of our approach to previous works coupling demonstrations with reinforcement learning (RL), and to goal-conditioned RL.

**Few-Shot Imitation by Behavior Cloning (BC).** Most previous works on few-shot imitation train policies with BC, using an action-prediction loss such as mean squared error or cross-entropy. Most of these works use policies that take demonstrations as input, as in DCRL. Some apply such policies to a new task without additional training (Duan et al., 2017; Yu et al., 2018; James et al., 2018; Huang et al., 2019; Dasari & Gupta, 2020; Bonardi et al., 2020). Others use *meta-learning*, which requires additional fine-tuning when presented with a new task at meta-test time (Finn et al., 2017b; Cachet et al., 2020), or exploit information about success or failure of trials of the policy on a new task (Zhou et al., 2020; Singh et al., 2020).

In contrast to DCRL, most methods exploiting BC require actions in the demonstrations, although some recent methods can cope without, for instance by inferring actions from state-only demonstrations (Guo et al., 2019). Also, while DCRL directly handles a domain shift between the demonstration and the agent, which is important when a robot should imitate a human, BC methods must be combined with special-purpose techniques to cope with domain shift, such as training using pairs of human and robot demonstrations (Bonardi et al., 2020). Finally, while DCRL can potentially improve upon suboptimal demonstrations, as it trains with reward functions, the performance of BC approaches is usually upper-bounded by the performance of the demonstrator. Nevertheless, Brown et al. (2019) recently proposed a method involving adding noise to demonstrations, which can improve on demonstrations in certain cases.

**Few-Shot Imitation by IRL.** Another approach to few-shot imitation is to infer a reward function from some demonstrations, a task known as *inverse reinforcement learning* or IRL (Ng et al., 1999), and then to learn a policy suitable for that reward. Yu et al. (2019a) adapt adversarial imitation learning (Ho & Ermon, 2016) to the one-shot setting and use a reward inferred from a discriminator, while Goo & Niekum (2019) use a reward based on classifying which of two observations comes later in the successful performance of a task. Such IRL-based methods can cope without actions in the demonstrations. Indeed, adversarial imitation learning can be seen as minimizing a divergence between the state-occupancy densities of the agent and the demonstrator — an objective that does not involve the demonstrator's actions.

While both DCRL and IRL-based methods train policies to maximize cumulative rewards, IRL-based methods rely on inferred reward functions, so they only have limited scope for improving upon suboptimal demonstrators, for instance by placing a strong prior on the reward function. Furthermore, existing IRL approaches assume that the structure and dynamics of the environment do not change from one task to another. To address such changes, Yu et al. (2019a) resort to additional exploration at test time, requiring up to 40 hours per test task. In future, it would be interesting to devise IRL algorithms that overcome this major limitation, as discussed in the Supplementary Material. DCRL overcomes this limitation: our results show that it copes with task-dependent transitions, without any training at test time.

**Multi-Task Imitation.** Arora et al. (2020) consider representation learning for multi-task imitation. As in multi-task learning (Sener & Koltun, 2018), they use architectures with both task-specific and task-independent components, rather than policies that take demonstrations as input. The authors provide theoretical guarantees, justifying the statistical benefit of representation learning, for settings with and without actions in demonstrations. It would be interesting to see if such guarantees can be extended to settings like those considered in this paper, where some reward functions are available for representation learning or where there may be a domain shift between the demonstrator and agent.

**Multi-Task Reinforcement Learning (MTRL).** MTRL is the application of reinforcement learning to multiple tasks jointly (Taylor & Stone, 2009; Lazaric, 2012; D'Eramo et al., 2020). Many MTRL methods aim to find a single policy that takes an encoding of the task as input (Yu et al., 2020), which maximizes the average return over a set of training tasks. DCRL could be seen as a MTRL method whose task encoding is an embedding of a collection of demonstrations.

Most previous work on MTRL only considers performance on the training tasks, and although the task identity must sometimes be inferred from interaction with the environment (Guo et al., 2020), it is usually given (Yu et al., 2019b; Xu et al., 2020). In contrast, DCRL aims to generalize to new test tasks. Works on MTRL that that do address generalization assume test-task rewards are available and that exploratory interaction with the test environment is possible (Rakelly et al., 2019; Li et al., 2020; Zhang et al., 2021). In contrast, DCRL requires neither test-task rewards, nor exploratory interaction with test environments.

MTRL often faces optimization difficulties, due to conflicting gradients and the need to balance optimisation between tasks, which are closely related to those encountered when trying to share actor and critic features in actor-critic methods (Cobbe et al., 2020). To overcome such difficulties, Teh et al. (2017) use task-specific policies that are constrained to remain close to a shared policy, Yu et al. (2020) perform 'gradient surgery' whenever gradients from two tasks conflict in the sense that they point away from one another, and Yang et al. (2020) explore a novel 'soft modularization' architecture. Such ideas offer a promising avenue for future improvement of our current implementation of DCRL.

**Demonstrations in RL.** Some works are similar to ours in the sense that they exploit demonstrations and maximize cumulative rewards, but they differ as they only address a single task at a time rather than doing few-shot imitation. Judah et al. (2014) maximize the return corresponding to a "shaping" reward function, subject to a constraint on a BC loss for a given set of demonstrations. Borsa et al. (2017) consider settings where an agent and an expert are simultaneously present in a common environment and the agent has access to the expert's current state. Such settings might lead to blocking situations, for instance the agent and expert might simultaneously try to pick up the same object.

Meanwhile, the deep $Q$-learning from demonstrations (DQfD) algorithm of Hester et al. (2018) and the normalized actor-critic (NAC) algorithm of Gao et al. (2018) use demonstrations not as inputs to a policy, but rather as experience in a replay buffer, with the aim of accelerating RL. This can be particularly helpful in problems with sparse reward functions. Similarly, soft $Q$ imitation learning (SQIL) also populates the replay buffer with demonstrations (Reddy et al., 2020), but whereas DQfD and NAC require a reward to be specified, SQIL uses a reward of $r = 1$ when the agent matches the demonstrated action in a demonstrated state, and a reward of $r = 0$ otherwise. SQIL attempts to overcome the error compounding problems of BC, and performs competitively with GAIL, without requiring adversarial training or the learning of a reward function.

Finally, several works have extended GAIL by adding an extra term to GAIL's reward. Bhattacharyya et al. (2019) add a term discouraging undesirable actions like braking too hard, while Huang et al. (2021) regress a reward term from human feedback and show that their algorithm can improve

upon suboptimal demonstrations, unlike GAIL.

**Goal-Conditioned Learning.** In *goal-conditioned reinforcement learning* (GCRL), the aim is to learn a policy that maps state observations and a "goal" to a distribution over actions, so as to maximize a cumulative reward (Nair et al., 2018; Ghosh et al., 2019; Nasiriany et al., 2019). Meanwhile, Ding et al. (2019) consider *goal-conditioned imitation learning*, proposing both a goal-conditioned BC algorithm and a goal-conditioned generative adversarial imitation algorithm based on Ho & Ermon (2016).

One way to compare GCRL with DCRL is to look at the information content carried by goals and demonstrations. A goal is possibly *more* informative as it is a clearly-defined set and reaching this set implies success, whereas the meaning of success may not always be clear from a demonstration, which is one reason for using few-shot rather than one-shot imitation. A goal is *less* informative as it relates to the state at a single time, whereas a demonstration is a time series. Thus, while goals can tell us *what* to achieve, demonstrations can also tell us *how* one could achieve it. Demonstrations can even express tasks with no goal state, such as the task of spinning a hula hoop.

## 3. Method

In this section, we formally present the few-shot imitation learning problem and the DCRL method. Then, we describe a specific family of demonstration-conditioned policy network architectures, that exploits axial attention to efficiently process multiple input demonstrations.

### 3.1. Preliminaries

We consider a setting in which tasks are sampled from a distribution of tasks $\eta$. Each task $\mu \sim \eta$ is associated with a Markov decision process $\mathcal{M}_\mu$ and a distribution over collections of demonstrations $\mathcal{D}_\mu$, as we now describe.

**Process $\mathcal{M}_\mu$.** Let $\Delta(\mathcal{X})$ denote the set of probability distributions over a set $\mathcal{X}$. The Markov decision process $\mathcal{M}_\mu := (\mathcal{S}, \mathcal{A}, \rho_\mu, P_\mu, R_\mu, \gamma)$ has set of states $\mathcal{S}$, set of actions $\mathcal{A}$, initial-state distribution $\rho_\mu \in \Delta(\mathcal{S})$, transition distribution $P_\mu : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, random reward $R_\mu$ which is a mapping from $\mathcal{S} \times \mathcal{A}$ to the space of real-valued random variables, and discount factor $\gamma \in [0, 1]$. We could define $\mathcal{S} := \cup_\mu \mathcal{S}_\mu$ or $\mathcal{A} := \cup_\mu \mathcal{A}_\mu$ if these sets depended on $\mu$. We consider both infinite-horizon settings with $\gamma < 1$, and episodic settings with $\gamma \leq 1$ which we model by assuming that some states are absorbing and provide zero reward.

Let $\mathcal{H} := \{(s_0, a_0, \dots, a_{t-1}, s_t) : s_i \in \mathcal{S}, i \leq t, a_j \in \mathcal{A}, j < t, t \in \mathbb{Z}_{\geq 0}\}$ be the space of state-action histories, and let $\Pi = \{\pi : \mathcal{H} \to \Delta(\mathcal{A})\}$ be the space of policies which map such histories to distributions over actions. Let the expected return of policy $\pi \in \Pi$ on task $\mu$ be $J_\mu(\pi) = \mathbb{E}_{\mu,\pi}[\sum_{t=0}^{\infty} \gamma^t R_\mu(s_t, a_t)]$, where $\mathbb{E}_{\mu,\pi}$ is the expectation is over state-action sequences $(s_0, a_0, s_1, a_1, \dots)$ sampled from $\rho_\mu$, $P_\mu$ and $\pi$. For each task $\mu$, we assume that Markov decision process $\mathcal{M}_\mu$ is such that $J_\mu(\cdot)$ exists for any policy, and that an optimal policy exists that maximises $J_\mu(\cdot)$. Let $J^*_\mu$ be the expected return of such an optimal policy.

**Distribution $\mathcal{D}_\mu$.** A demonstration is a sequence $d := (o_0, o_1, \dots, o_{T-1})$ of observations $o_t \in \Omega$ of random length $T \in \mathbb{Z}_{>0}$. Observations might be state-action pairs so that $\Omega \subseteq \mathcal{S} \times \mathcal{A}$, they might be states so that $\Omega \subseteq \mathcal{S}$, or they might be images or some other sensor measurements that only provide partial information about the state. In general, such observations come from a demonstrator, they need not be associated with the Markov decision process $\mathcal{M}_\mu$. We denote the set of all observation sequences of finite non-zero length by $\Omega^+$, using the Kleene plus.

Collections of demonstrations of task $\mu$ are sampled from a distribution $\mathcal{D}_\mu$. A collection of demonstrations $\mathbf{d} := (d_0, \dots, d_{n-1}) \sim \mathcal{D}_\mu$ consists of a random number $n \in \mathbb{Z}_{>0}$ of individual demonstrations. We denote the set of collections of demonstrations by $\mathbf{D} := (\Omega^+)^+$.

**Problem Statement.** A *few-shot imitation problem* is given by a distribution $\eta$ over tasks, and for each task $\mu$ a Markov decision process $\mathcal{M}_\mu$ and a distribution $\mathcal{D}_\mu$ over collections of demonstrations, as described above. The aim is to find an agent $\alpha : \mathbf{D} \to \Pi$ which maps a collection of demonstrations $\mathbf{d} \sim \mathcal{D}_\mu$ of a task $\mu \sim \eta$ to a policy, so as to maximize the average return over tasks:

$$\max_{\alpha:\mathbf{D}\to\Pi} \mathbb{E}_{\mu\sim\eta} \mathbb{E}_{\mathbf{d}\sim\mathcal{D}_\mu} J_\mu(\alpha(\mathbf{d})).$$

It is customary to assume that a policy is given by a probability mass or density function over the actions $\mathcal{A}$. Under this assumption, the above aim is equivalent to finding a *demonstration-conditioned policy* $\pi$ that assigns probability $\pi(a|h, \mathbf{d})$ to action $a$, given history $h$ and demonstrations $\mathbf{d}$, which maximizes

$$\mathbb{E}_{\mu\sim\eta} \mathbb{E}_{\mathbf{d}\sim\mathcal{D}_\mu} \mathbb{E}_{\mu,\pi(\cdot|\cdot,\mathbf{d})} \left[ \sum_{t=0}^{\infty} \gamma^t R_\mu(s_t, a_t) \right]. \quad (1)$$

### 3.2. DCRL Method

DCRL takes a simple and direct approach to the few-shot imitation problem. As input, we are given a training set

$$\mathcal{X} := \{(\mathbf{d}^i, \mathcal{M}_{\mu^i})\}_{i=0}^{N-1},$$

where each $\mathbf{d}^i \in \mathbf{D}$ is a collection of demonstrations of a task $\mu^i$ and $\mathcal{M}_{\mu^i}$ is the Markov decision process for that task. We train a demonstration-conditioned policy $\pi$ to

**Algorithm 1** Demonstration-conditioned reinforcement learning

**Input:** Training set $\mathcal{X} := \{(\mathbf{d}^i, \mathcal{M}_{\mu^i})\}_{i=0}^{N-1}$, trainable parameters $\theta$
Buffer $\leftarrow \emptyset$
**while** not converged **do**
  $i \sim$ Uniform($\{0, \dots, N-1\}$)
  $s_0$, done $\leftarrow$ InitEnvironment($\mathcal{M}_{\mu^i}$)
  **for** $t = 0, 1, \dots$ **until** done **do**
    $a_t \sim \pi_\theta(\cdot|h_t, \mathbf{d}^i)$ where $h_t = (s_0, a_0, \dots, s_t)$
    $R_t, s_{t+1}$, done $\leftarrow$ EnvironmentStep($\mathcal{M}_{\mu^i}, s_t, a_t$)
    Buffer $\leftarrow$ Buffer $\cup \{(R_t, h_{t+1}, \text{done}, \mathbf{d}^i)\}$
  **end for**
  **if** it is time for an update **then**
    $\theta \leftarrow$ UpdatePolicy(Buffer, $\theta$)
  **end if**
**end while**
**return** $\pi_\theta$

maximize the empirical average cumulative reward

$$\frac{1}{N} \sum_{i=0}^{N-1} J_{\mu^i}(\pi(\cdot|\cdot, \mathbf{d}^i)). \tag{2}$$

To approximately maximize this objective, we propose to use an RL algorithm in which we simply append the demonstration to the state in the replay buffer, as shown in Algorithm 1.

At test time, DCRL does not require access to a reward function. Nor does it require additional interaction with the test environment to tune the policy. Rather, given a collection of demonstrations $\mathbf{d}^{\text{test}}$ of a new test task, one simply feeds those demonstrations along with the current history $h$ to the learned policy, and takes actions $a \sim \pi(\cdot|h, \mathbf{d}^{\text{test}})$.

**Generalization.** When might DCRL produce policies that generalize to new tasks $\mu$ which are not present in the training set?

For there to exist a policy that attains a high value for the objective of the few-shot imitation problem (1), the demonstrations must carry sufficient information about the nature of the task at hand. For instance, one might consider few-shot imitation problems that are *separable* in the sense that there exists a mapping $\alpha : \mathbf{D} \to \Pi$ that attains the upper bound $\mathbb{E}_{\mu\sim\eta} J^*_\mu$ on the objective (1).

In the next subsection, we shall explore demonstration-conditioned policies with a specific structure $\pi(a|h, \mathbf{d}) = F(a, h, \Phi(\mathbf{d}))$, where $\Phi : \mathbf{D} \to \mathcal{E}$ maps a collection of demonstrations to an embedding space $\mathcal{E}$, and $F$ maps histories and embeddings to action probabilities.

One might think of the embedding function $\Phi$ as a classifier that maps demonstrations to task identities, and $F$ as a policy for each identified task. However, different tasks may have identical optimal policies and are not always distinguishable based on demonstrations. In such situations, even if a perfect classifier mapping demonstrations to task identities does not exist, it is sometimes still possible to attain the upper bound

$$\mathbb{E}_{\mu\sim\eta}\mathbb{E}_{\mathbf{d}\sim\mathcal{D}_\mu} J_\mu(F(\cdot|\cdot, \Phi(\mathbf{d}))) = \mathbb{E}_{\mu\sim\eta} J^*_\mu$$

on the objective (1). In the Supplementary Material, we present experiments exploring the notion that two collections of demonstrations are close (in some sense) under $\Phi$ if and only if they have similar optimal policies, and that generalization to new tasks is achieved by interpolating in this embedding space.

### 3.3. Transformer-Based Policy Network

Motivated by the role of attention in human task control (Rothkopf et al., 2007) and its successful application to natural language processing (Vaswani et al., 2017), several previous works on few-shot imitation have explored policy architectures that rely on attention mechanisms and transformers (Duan et al., 2017; Mishra et al., 2018; James et al., 2018; Dasari & Gupta, 2020; Cachet et al., 2020). While Duan et al. (2017) mention the potential interest of policies that condition on multiple demonstrations to resolve potential ambiguities, all previously proposed policies either take only one demonstration as input, or they encode several demonstrations independently and then average the encodings (James et al., 2018). In contrast, we consider policies with *cross-demonstration attention*, which accept a variable number of demonstrations as input, and which process demonstrations simultaneously, enabling a richer integration of information than is possible by averaging. This advantage is particularly clear in our experiments on navigation (Section 4.2).

While the results of transformer-based policies are impressive, their computational and memory complexities grow quadratically with the size of their input. This becomes prohibitive when the input consists of multiple demonstrations, each of which is a multivariate time series. To overcome this cost, we explore the application of transformers with *axial attention*, which were recently proposed by Ho et al. (2019). Axial attention is one of several techniques (Tay et al., 2020) recently proposed to improve the efficiency of transformers. While axial attention has previously been applied to visual data (Wang & Liu, 2020), it has not previously been applied to policy architectures.

In the remainder of this section, we describe the overall architecture and then our encoder layers with axial attention.

**Overall Architecture.** Figure 2 shows our proposed policy architecture. It consists of an encoder and a decoder. The encoder maps a collection of demonstrations to an em-

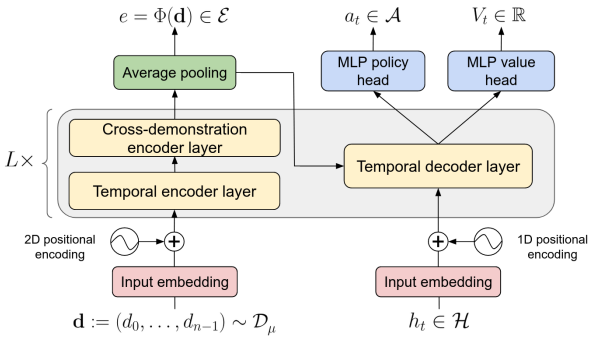


*Figure 2.* Simplified view of the proposed DCRL policy architecture. The model takes a collection of demonstrations and the agent's history as input, and outputs the action distribution (policy head) and estimated value function (value head). See the Supplementary Material for hyperparameters.

bedding. The decoder treats this embedding as context and maps the agent's history to an action and a value function.

In more detail, we are given a collection of demonstrations $\mathbf{d}$, which we treat as an array of size $T \times n \times d_{\text{obs}}$, where $T$ is the maximum length of the demonstrations, $n$ is the number of demonstrations, and $d_{\text{obs}}$ is the dimension of the observations. Demonstrations shorter than $T$ are masked, and we map each observation to a latent space of dimension $H$. As multi-head self-attention is equivariant to permutations of its input, we add 2D positional encoding to the demonstrations, as recently proposed for images (Wang & Liu, 2020; Carion et al., 2020). We pass the result through a sequence of $L$ encoder layers, each encoder layer having distinct trainable parameters, but an identical architecture, which is described below. The output of each encoder layer is of size $T \times n \times H$. We average this output over its demonstration dimension, to get an embedding $e$ of size $T \times H$.

Next we process the agent's history $h_t$. We map each element of $h_t$ to a latent space of dimension $H$, and we add 1D positional encoding. The resulting array of size $T' \times H$, where $T'$ is the length of the history, is fed through a sequence of $L$ decoder layers. Each decoder layer has the architecture used in Vaswani et al. (2017), consisting of multi-head self-attention, multi-head cross-attention using embedding $e$, and a feedforward network, each surrounded by a residual connection and followed by layer normalization. The output of the last decoder layer is fed to a multi-layer perceptron, giving a distribution from which we sample an action $a_t$. Optionally, depending on the RL algorithm, the output is also fed to a second multi-layer perceptron, giving an estimate of the state-value function $V_t$. Full details can be found in the Supplementary Material.

**Encoder Layers with Axial Attention.** Following the axial attention approach of Ho et al. (2019), each of our encoder layers consists of a temporal layer, followed by a cross-demonstration layer, and then a pointwise feedforward network. Each of these layers is surrounded by a residual connection and followed by layer normalization. For input $X \in \mathbb{R}^{T \times n \times H}$, the temporal layer has output $Y \in \mathbb{R}^{T \times n \times H}$ with elements

$$Y_{tik} = \text{MultiHeadSelfAttention}(X^{(0,i)})_{tk},$$

where each $X^{(0,i)} \in \mathbb{R}^{T \times H}$ is the matrix with elements $X^{(0,i)}_{tk} = X_{tik}$, and the multi-head self-attention is that of Vaswani et al. (2017). The cross-demonstration layer has output $Y \in \mathbb{R}^{T \times n \times H}$ with elements

$$Y_{tik} = \text{MultiHeadSelfAttention}(X^{(1,t)})_{ik},$$

where each $X^{(1,t)} \in \mathbb{R}^{n \times H}$ has elements $X^{(1,t)}_{ik} = X_{tik}$.

For inputs in $\mathbb{R}^{T \times n \times H}$, the computation and (backpropagation) memory complexities of a conventional encoder are both $O(T^2 n^2)$, considering $H$ as well as the number of heads and layers to be fixed, whereas with axial attention, these complexities are reduced to $O(Tn(T+n))$. Without this saving, we began to run out of GPU memory while training, even for $n = 4$.

## 4. Experiments

This section addresses the following questions. **(Q1)** Is DCRL a competitive approach to few-shot imitation? **(Q2)** Does DCRL provide a good initialisation for fine-tuning on test tasks? **(Q3)** What is the benefit of cross-demonstration attention? **(Q4)** How robust is DCRL to a domain shift between the demonstrator and the agent? **(Q5)** Can DCRL improve upon suboptimal demonstrations?

First, we describe the baseline methods with which we compare, and our evaluation protocol. Then, we discuss our results on a robotic manipulation benchmark and a navigation benchmark.

**Baselines.** We compare DCRL with two *demonstration-conditioned behavioural cloning* (DCBC) methods, and a multi-task reinforcement learning (MTRL) approach. Each DCBC method uses the same architecture as DCRL (the MLP value head in Figure 2 is simply ignored), but is trained to minimize a BC loss. For continuous actions, the BC loss is the squared error in the mean action output of the policy MLP, and for discrete actions, it is the cross-entropy loss.

*DCBC+MT (Multi-Task).* This baseline minimizes the BC loss for predicting actions in the training demonstrations.

*DCBC+REPTILE.* This baseline uses Reptile coupled with BC loss to meta-train the model. Reptile (Nichol et al., 2018) is a meta-learning algorithm that yields similar performance to MAML (Finn et al., 2017a), while being less computationally expensive.

Except for the architecture and use of multiple demonstrations as input, DCBC+MT is the approach of Duan et al. (2017) and DCBC+REPTILE is that of Cachet et al. (2020).

*MTRL.* This baseline pretrains a model with multi-task RL, to maximize the average reward over training tasks, as in DCRL. However, we ablated the encoder part of the network, which takes the demonstrations as input. When training, we provide the task ID to the last layer, so the other layers learn a task-independent representation. At test time, we randomly initialize the last layer, and finetune it with BC, while freezing the rest of the network.

The DCRL, MTRL and DCBC-based baselines take the agents' full observation-action histories as input. History-dependent policies are a natural choice here, as the benchmark environments are both POMDPs (Igl et al., 2018), and because some tasks (e.g. clap twice then stop) are hard to imitate given only an agent's current state.

**Training Protocol.** First, we sample 5000 demonstrations of each task. To do so, we train one policy per task with PPO (Schulman et al., 2017), until each policy has at least a 99% success rate according to a task-specific success criterion. We then sample successful trajectories from these policies. We train DCRL and the two baselines using demonstrations sampled uniformly from this collection, and sample the number of such demonstrations uniformly from $\{1, \dots, 4\}$.

We train DCRL as shown in Algorithm 1, using PPO in line 12. We chose PPO for its simplicity, its relatively short training times, and the high quality of the resulting policies. Most other RL algorithms could be used instead. It takes about one day to train DCRL for both benchmarks, using a Tesla V100 GPU. Training requires about 250 million environment frames. The model has about $5.5 \times 10^5$ learnable parameters.

**Evaluation Protocol.** We evaluate the methods on tasks not present in the dataset. To evaluate a policy for a single task, we apply it for 300 episodes, with randomly sampled demonstrations and initial conditions for each episode. We report the average and standard deviation of the return and success rate for each task in the Supplementary Material. For brevity, this section mostly reports averages over all test tasks and focuses on success rates, as these can be interpreted without a detailed understanding of the reward.

### 4.1. Manipulation Benchmark

We use Meta-World, a robotic manipulation benchmark, originally designed to assess the performance of meta-learning algorithms (Yu et al., 2019b).

The reward function specified in that paper sometimes makes it preferable for agents to stay in regions of high reward, than to successfully complete a task. So in this paper, we use a modified reward, which acts like the time derivative of the original reward. Our task-specific policies always converge to a 100% success rate for this reward, as defined by the Meta-World success criteria, which we did not modify. Thus, our modified reward preserves the behaviour originally intended by the environment designer. The modified reward is discussed in detail in the Supplementary Material.

Otherwise, the benchmark is as in Yu et al. (2019b), with observations in $\mathbb{R}^{13}$, actions in $[-1, 1]^4$ and transitions computed with MuJoCo (Todorov et al., 2012). The positions of the objects manipulated and the goal are sampled at the start of each episode, and an episode ends if a task-specific success criterion is met, or if the time-out of 200 steps (Yu et al., 2019c) is reached.

We adopt the evaluation protocol **ML45** defined in Yu et al. (2019b). Thus, we randomly partition the 50 tasks into 10 groups of 5. Then we work with 10 models for each method compared, each model being trained on 45 tasks and tested on 5 hold-out tasks.

On this benchmark, using an Nvidia 2080 Ti GPU, the execution time of our transformer-based architecture is as follows. Embedding a collection of four demonstrations takes 3.05 ms, and this is only done once per episode. Predicting an action from the embedding and history takes up to 2.7 ms.

#### 4.1.1. Few-Shot Imitation

*Table 1.* Return and success rate of DCRL and DCBC+MT averaged over all Meta-World tasks, for one or five input demonstrations.

| | Return | | Success Rate | |
|---|---|---|---|---|
| # Demonstrations | 1 | 5 | 1 | 5 |
| DCRL (ours) | **296.34** | **289.21** | **48%** | **48%** |
| DCBC+MT | 109.25 | 120.42 | 17% | 24% |

Table 1 compares the discounted return and success rate of DCRL with those of DCBC+MT, on the task of few-shot imitation, with no fine-tuning. DCRL significantly improves on the baseline, although its performance does not increase when more demonstrations are provided.

#### 4.1.2. Few-Shot Imitation with Fine-Tuning

We now ask if we can fine-tune DCRL effectively on new tasks, assuming that actions are available in the demonstrations. Fine-tuning DCRL with behaviour cloning only takes few seconds for each task.

As shown in Figure 3, DCRL attains a 90% success rate over all Meta-World tasks, after fine-tuning on only four demonstrations. This is a large improvement on the success rates of $RL^2$, PEARL and MAML reported in Yu et al. (2019b), as discussed in the Supplementary Material.

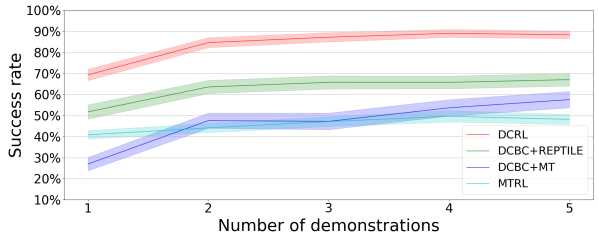


*Figure 3.* Success rate averaged over all Meta-World tasks, after fine-tuning with a given number of demonstrations, then using those same demonstrations as policy input.

As in Cachet et al. (2020), we observe that DCRL+REPTILE gives a better initialisation point than DCBC+MT, although this advantage decreases with the number of demonstrations. MTRL performs comparably to DCBC+MT. Meanwhile, DCRL gives the best initialisation point. We conjecture that this is because DCRL can interact with the environment during training, and thus overcome the compounding errors that plague BC-based methods (Rajaraman et al., 2020).

#### 4.1.3. Robustness to Domain Shift

To assess the robustness of DCRL to a domain shift between the demonstrator and the agent, we collected demonstrations using PPO policies for a LIMS2-AMBIDEX robot (Kim, 2015), rather than the Sawyer robot used in the original Meta-World benchmark. The AMBIDEX manipulator has seven degrees of freedom like the Sawyer, but its observations are in $\mathbb{R}^{18}$ as they contain information about the gripper orientation, and it has a different mechanical structure. We were only able to get PPO to converge for the AMBIDEX model on 43 out of the 50 Meta-World tasks, so the results of this experiment are averages over those tasks.

Table 2 presents the average return and success rate for DCRL with no domain shift (trained and tested with Sawyer demonstrations and Sawyer environment), and for DCRL with domain shift (trained and tested with AMBIDEX demonstrations and Sawyer environment). The results for the two settings are similar, suggesting that DCRL can indeed cope with a domain shift. In future, it will be interesting to explore more extreme domain shifts, involving natural language and videos of humans.

*Table 2.* Return and success rate of DCRL averaged over 43 Meta-World tasks, using demonstrations from Sawyer (top row) or from AMBIDEX (bottom row) as input.

| | Return | | Success Rate | |
|---|---|---|---|---|
| # Demonstrations | 1 | 5 | 1 | 5 |
| DCRL | 315.90 | 322.69 | 51% | 51% |
| DCRL (AMBIDEX) | 308.42 | 328.70 | 45% | 48% |

#### 4.1.4. Improving on Suboptimal Demonstrations

To explore if DCRL can outperform a suboptimal demonstrator when presented with a new task, we sampled demonstrations by adding noise to the actions taken by task-specific expert PPO policies. We added zero-mean Gaussian noise with covariance $\sigma^2 I_{4\times4}$, where the standard deviation $\sigma$ is an adjustable parameter.

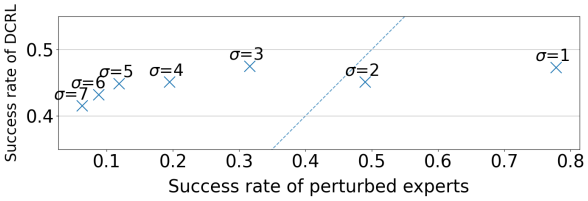


*Figure 4.* Success rate of perturbed experts against success rate of DCRL using three demonstrations from the perturbed experts. Success is averaged over all Meta-World tasks.

Figure 4 shows that the experts still outperform DCRL for $\sigma \leq 2$. However, for $\sigma > 2$, DCRL is clearly more successful than the task-specific demonstrator, even though it has never encountered the task before.

### 4.2. Navigation Benchmark

Our second benchmark involves 60 tasks, each corresponding to a maze layout. As shown in Figure 5, the agent must navigate between a given pair of positions. Our main motivation for introducing this benchmark is to challenge agents' abilities to integrate information across demonstrations, to a greater extent than the Meta-World benchmark. As no information in an agent's observation specifies the current layout, the agents must use the demonstrations to learn about the layout in order to reach its goal point efficiently without hitting walls.

In each task, observations are in $\mathbb{R}^7$ (agent and goal positions, agent velocity and orientation), there are four actions (forward, backward, turn left and turn right), the reward is minus the Euclidean distance between the agent and the goal with a bonus for reaching the goal and a penalty for hitting walls, the transition function is computed with ViZDoom (Kempka et al., 2016), and the initial position of the agent and the goal are sampled uniformly. Full details are in the Supplementary Material.

In all experiments on this benchmark, we train on a fixed set of 50 mazes and test on the remaining 10 mazes.

#### 4.2.1. Few-Shot Imitation

Table 3 compares the performance of DCRL with DCBC+MT, for few-shot imitation with no fine-tuning. As in the

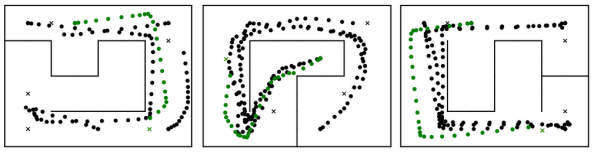

*Figure 5.* Three examples of navigation tasks. Crosses are goals, black dots are demonstrations and green dots are a DCRL agent's history.

analogous results for Meta-World (Table 1), DCRL significantly improves on the baseline, but in contrast to those analogous results, DCRL's performance increases as more demonstrations are provided. One explanation for this increase may be that different demonstrations often cover different parts of the maze, so some pairs of initial and goal positions may only be addressed by integrating information from different demonstrations.

*Table 3.* Return and success rate averaged over the 10 test mazes using one or five demonstrations as input.

| | Return | | Success Rate | |
|---|---|---|---|---|
| # Demonstrations | 1 | 5 | 1 | 5 |
| DCRL | **61.79** | **74.17** | **77%** | **85%** |
| DCBC+MT | 17.52 | 17.87 | 68% | 68% |

#### 4.2.2. Few-Shot Imitation with Fine-Tuning

Figure 6 shows that fine-tuning provides smaller performance improvements for this benchmark than the analogous results for Meta-World (Figure 3). Our interpretation of this is that most demonstrations are from episodes with rather different initial and goal positions than those in the episode that the agent is confronted with. Thus, while the agent can learn about the structure of the maze by examining the demonstrations, it does not stand to benefit much from cloning the demonstrator's actions.

The MTRL baseline lags further behind other methods here than it did in the manipulation benchmark. Partly, this is because it lacks the cross-demonstration attention available to the other methods. Moreover in MTRL, task-specific information only influences the last layer of the policy network, whereas in the demonstration-conditioned methods, the influence is much richer, with every decoder layer accessing the task embedding via cross-attention.

#### 4.2.3. Cross-Demonstration Attention

To understand the benefit of cross-demonstration attention, we compared DCRL using five demonstrations as input, with an algorithm in which we feed each of these five demonstrations to DCRL one at a time, and then we average the resulting action probabilities. As shown in Figure 7, cross-

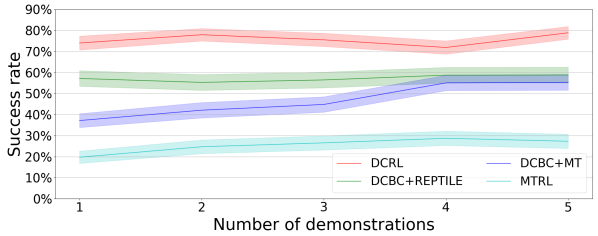


*Figure 6.* Success rate averaged over 10 unseen mazes using the given number of demonstrations both as fine-tuning data and as input.

demonstration attention has a consistent advantage for all 10 test mazes, in line with the expectation of Duan et al. (2017) that attending to multiple demonstrations should help when one demonstration does not fully resolve ambiguity in the objective.

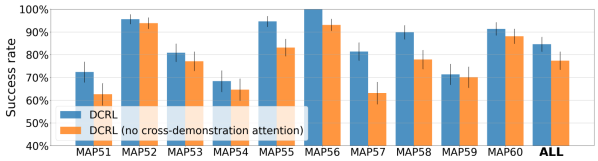


*Figure 7.* Success rate for each test maze using 5 demonstrations, with and without cross-demonstration attention.

## 5. Conclusions

The aim of few-shot imitation is to reduce the number of demonstrations required to learn to perform new tasks. We presented a new approach to few-shot imitation called demonstration-conditioned reinforcement learning (DCRL). While DCRL requires the specification of reward functions for training, we show that this extra cost can be outweighed by a reduction in the number of demonstrations required at inference time and an improved success rate on new tasks, relative to other recent few-shot imitation methods. Moreover, our results on robotic manipulation and navigation benchmarks show that DCRL can improve on suboptimal demonstrators and succeed even when there is a domain shift between the agent and demonstrator.

## Acknowledgements

We thank Taeyoon Lee and Keunjun Choi for the LIMS2-AMBIDEX model, Seungsu Kim, Tomi Silander, Diane Larlus and the reviewers for their many helpful comments, as well as Sangok Seok for his support and encouragement.

## References


Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning, ICML*, 2004.

Arora, S., Du, S. S., Kakade, S., Luo, Y., and Saunshi, N. Provable representation learning for imitation learning via bi-level optimization. In *International Conference on Machine Learning, ICML*, 2020.

Bhattacharyya, R. P., Phillips, D. J., Liu, C., Gupta, J. K., Driggs-Campbell, K. R., and Kochenderfer, M. J. Simulating emergent properties of human driving behavior using multi-agent reward augmented imitation learning. In *International Conference on Robotics and Automation, ICRA*, pp. 789–795, 2019.

Bonardi, A., James, S., and Davison, A. J. Learning one-shot imitation from humans without humans. *IEEE Robotics and Automation Letters*, 5(2):3533–3539, 2020.

Borsa, D., Piot, B., Munos, R., and Pietquin, O. Observational learning by reinforcement learning. *arXiv preprint arXiv:1706.06617*, 2017.

Boyd, R., Richerson, P. J., and Henrich, J. The cultural niche: Why social learning is essential for human adaptation. *Proceedings of the National Academy of Sciences*, 108:10918–10925, 2011.

Brown, D. S., Goo, W., and Niekum, S. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on Robot Learning, CoRL*, 2019.

Cachet, T., Perez, J., and Kim, S. Transformer-based meta-imitation learning for robotic manipulation. In *3rd Robot Learning Workshop at NeurIPS*, 2020.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In *European Conference on Computer Vision, ECCV*, volume 12346 of *Lecture Notes in Computer Science*, pp. 213–229, 2020.

Charpentier, C. J., Iigaya, K., and O'Doherty, J. P. A neuro-computational account of arbitration between choice imitation and goal emulation during human observational learning. *Neuron*, 106(4):687–699, 2020.

Cobbe, K., Hilton, J., Klimov, O., and Schulman, J. Phasic policy gradient. *arXiv:2009.04416*, 2020.

Dasari, S. and Gupta, A. Transformers for one-shot visual imitation. In *Conference on Robot Learning, CoRL*, 2020.

D'Eramo, C., Tateo, D., Bonarini, A., Restelli, M., and Peters, J. Sharing knowledge in multi-task deep reinforcement learning. In *International Conference on Learning Representations, ICLR*, 2020.

Ding, Y., Florensa, C., Abbeel, P., and Phielipp, M. Goal-conditioned imitation learning. In *Advances in Neural Information Processing Systems, NeurIPS*, pp. 15298–15309, 2019.

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

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135. PMLR, 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.

Gao, Y., Xu, H., Lin, J., Yu, F., Levine, S., and Darrell, T. Reinforcement learning from imperfect demonstrations. In *ICLR Workshop Track*, 2018.

Ghosh, D., Gupta, A., and Levine, S. Learning actionable representations with goal conditioned policies. In *International Conference on Learning Representations, ICLR*, 2019.

Goo, W. and Niekum, S. One-shot learning of multi-step tasks from observation via activity localization in auxiliary video. In *International Conference on Robotics and Automation, ICRA*, pp. 7755–7761, 2019.

Guo, X., Chang, S., Yu, M., Tesauro, G., and Campbell, M. Hybrid reinforcement learning with expert state sequences. In *AAAI Conference on Artificial Intelligence, AAAI*, pp. 3739–3746, 2019.

Guo, Z. D., Pires, B. Á., Piot, B., Grill, J., Altché, F., Munos, R., and Azar, M. G. Bootstrap latent-predictive representations for multitask reinforcement learning. In *International Conference on Machine Learning, ICML*, 2020.

Hester, T., Vecerík, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., Dulac-Arnold, G., Agapiou, J. P., Leibo, J. Z., and Gruslys, A. Deep Q-learning from demonstrations. In *AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Ho, J. and Ermon, S. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems, NIPS*, pp. 4565–4573, 2016.

Ho, J., Kalchbrenner, N., Weissenborn, D., and Salimans, T. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

Hoehl, S., Keupp, S., Schleihauf, H., McGuigan, N., Buttelmann, D., and Whiten, A. 'Over-imitation': A review and appraisal of a decade of research. *Developmental Review*, 51:90–108, 2019.

Huang, D., Nair, S., Xu, D., Zhu, Y., Garg, A., Fei-Fei, L., Savarese, S., and Niebles, J. C. Neural task graphs: Generalizing to unseen tasks from a single video demonstration. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pp. 8565–8574, 2019.

Huang, J., Juan, R., Gomez, R., Nakamura, K., Sha, Q., He, B., and Li, G. GAN-based interactive reinforcement learning from demonstration and human evaluative feedback. *arXiv:2104.06600*, 2021.

Igl, M., Zintgraf, L. M., Le, T. A., Wood, F., and Whiteson, S. Deep variational reinforcement learning for POMDPs. In *International Conference on Machine Learning, ICML*, 2018.

James, S., Bloesch, M., and Davison, A. J. Task-embedded control networks for few-shot imitation learning. In *Conference on Robot Learning, CoRL*, pp. 783–795, 2018.

Judah, K., Fern, A., Tadepalli, P., and Goetschalckx, R. Imitation learning with demonstrations and shaping rewards. In Brodley, C. E. and Stone, P. (eds.), *AAAI Conference on Artificial Intelligence*, pp. 1890–1896, 2014.

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaśkowski, W. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games, CIG*, pp. 1–8, 2016.

Kim, Y. Design of low inertia manipulator with high stiffness and strength using tension amplifying mechanisms. *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 5850–5856, 2015.

Lazaric, A. Transfer in reinforcement learning: A framework and a survey. In *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, pp. 143–173. Springer, 2012.

Li, J., Vuong, Q., Liu, S., Liu, M., Ciosek, K., Christensen, H. I., and Su, H. Multi-task batch reinforcement learning with metric learning. In *Advances in Neural Information Processing Systems, NeurIPS*, 2020.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. A simple neural attentive meta-learner. In *International Conference on Learning Representations, ICLR*, 2018.

Nair, A., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems, NeurIPS*, pp. 9209–9220, 2018.

Nasiriany, S., Pong, V., Lin, S., and Levine, S. Planning with goal-conditioned policies. In *Advances in Neural Information Processing Systems, NeurIPS*, pp. 14814–14825, 2019.

Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning, ICML*, pp. 278–287. Morgan Kaufmann, 1999.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Pomerleau, D. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1): 88–97, 1991.

Rajaraman, N., Yang, L., Jiao, J., and Ramchandran, K. Toward the fundamental limits of imitation learning. In *Advances in Neural Information Processing Systems, NeurIPS*, 2020.

Rakelly, K., Zhou, A., Finn, C., Levine, S., and Quillen, D. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International Conference on Machine Learning, ICML*, 2019.

Reddy, S., Dragan, A. D., and Levine, S. SQIL: Imitation learning via reinforcement learning with sparse rewards. In *International Conference on Learning Representations, ICLR*, 2020.

Ross, S. and Bagnell, D. Efficient reductions for imitation learning. In *International Conference on Artificial Intelligence and Statistics, AISTATS*, pp. 661–668, 2010.

Rothkopf, C., Ballard, D., and Hayhoe, M. Task and context determine where you look. *Journal of Vision*, 7(14), 2007.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sener, O. and Koltun, V. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems, NeurIPS*, pp. 527–538, 2018.

Singh, A., Jang, E., Irpan, A., Kappler, D., Dalal, M., Levine, S., Khansari, M., and Finn, C. Scalable multi-task imitation learning with autonomous improvement. In *IEEE International Conference on Robotics and Automation, ICRA*, pp. 2167–2173, 2020.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*, 2020.

Taylor, M. and Stone, P. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

Teh, Y. W., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., and Pascanu, R. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems, NeurIPS*, pp. 4496–4506, 2017.

Todorov, E., Erez, T., and Tassa, Y. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 5026–5033, 2012.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems, NIPS*, pp. 5998–6008, 2017.

Venuto, D., Boussioux, L., Wang, J., Dali, R., Chakravorty, J., Bengio, Y., and Precup, D. Avoidance learning using observational reinforcement learning. *arXiv preprint arXiv:1909.11228*, 2019.

Wang, Z. and Liu, J.-C. Translating math formula images to latex sequences using deep neural networks with sequence-level training. *International Journal on Document Analysis and Recognition, IJDAR*, pp. 1–13, 2020.

Xu, Z., Wu, K., Che, Z., Tang, J., and Ye, J. Knowledge transfer in multi-task deep reinforcement learning for continuous control. In *Advances in Neural Information Processing Systems, NeurIPS*, 2020.

Yang, R., Xu, H., Wu, Y., and Wang, X. Multi-task reinforcement learning with soft modularization. In *Advances in Neural Information Processing Systems, NeurIPS*, 2020.

Yu, L., Yu, T., Finn, C., and Ermon, S. Meta-inverse reinforcement learning with probabilistic context variables. In *Advances in Neural Information Processing Systems, NeurIPS*, pp. 11749–11760, 2019a.

Yu, T., Finn, C., Dasari, S., Xie, A., Zhang, T., Abbeel, P., and Levine, S. One-shot imitation from observing humans via domain-adaptive meta-learning. In *Robotics: Science and Systems, RSS*, 2018.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning, CoRL*, 2019b.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., Levine, S., Pong, V., et al. Meta-World source code. https://github.com/rlworkgroup/metaworld, 2019c.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems, NeurIPS*, 2020.

Zhang, A., Sodhani, S., Khetarpal, K., and Pineau, J. Learning robust state abstractions for hidden-parameter block MDPs. In *International Conference on Learning Representations, ICLR*, 2021.

Zhou, A., Jang, E., Kappler, D., Herzog, A., Khansari, M., Wohlhart, P., Bai, Y., Kalakrishnan, M., Levine, S., and Finn, C. Watch, try, learn: Meta-learning from demonstrations and rewards. In *International Conference on Learning Representations, ICLR*, 2020.