SQUIRL: Robust and Efficient Learning from Video Demonstration of Long-Horizon Robotic Manipulation Tasks

Bohan Wu, Feng Xu, Zhanpeng He, Abhi Gupta, and Peter K. Allen

Abstract— Recent advances in deep reinforcement learning (RL) have demonstrated its potential to learn complex robotic manipulation tasks. However, RL still requires the robot to collect a large amount of real-world experience. To address this problem, recent works have proposed learning from expert demonstrations (LfD), particularly via inverse reinforcement learning (IRL), given its ability to achieve robust performance with only a small number of expert demonstrations. Nevertheless, deploying IRL on real robots is still challenging due to the large number of *robot* experiences it requires. This paper aims to address this scalability challenge with a robust, sample-efficient, and general meta-IRL algorithm, SQUIRL, that performs a new but related long-horizon task robustly given only a single video demonstration. First, this algorithm bootstraps the learning of a task encoder and a task-conditioned policy using behavioral cloning (BC). It then collects real-robot experiences and bypasses reward learning by *directly* recovering a Q-function from the combined robot and expert trajectories. Next, this algorithm uses the Q-function to re-evaluate all cumulative experiences collected by the robot to improve the policy quickly. In the end, the policy performs more robustly (90%+ success) than BC on new tasks while requiring no trial-and-errors at test time. Finally, our real-robot and simulated experiments demonstrate our algorithm's generality across different state spaces, action spaces, and vision-based manipulation tasks, e.g., pick-pour-place and pick-carry-drop.

I. INTRODUCTION

We aspire robots to become generalists who acquire new complex skills robustly and quickly. The robotic system, whether planned or learned, needs to leverage its existing knowledge to solve a new but related task in an efficient yet high-performance manner. Thanks to recent advances in machine learning and sim-to-real transfer mechanisms, short-horizon robotic manipulation such as grasping has improved in performance. However, many real-world robotic manipulation tasks are long-horizon, diverse, and abundant in volume. In the absence of a scalable and systematic way to construct simulation environments for a large number of tasks, the robot needs to learn a new task directly in the physical world from only a handful of trials, due to the high cost of collecting real-robot trial-and-errors and experiences.

We observe that real-world robotic skill acquisition can become more sample-efficient in several important ways. First, we notice that humans learn tasks quickly by watching others perform similar tasks. Among many forms of task representations such as rewards, goal images, and language instructions, human demonstrations guide exploration effectively and can lead to significant sample efficiency gains. Furthermore, learning from video demonstrations sidesteps

Train: 117 Video Demonstrations across 117 Tasks + 90 Robot Trials

| Approach | Pick | Carry | Pour Green | Carry | Place |

| Approach | Pick | Carry | Pour Yellow | Carry | Place |

90 Real-robot Trial-and-errors (Practices)

SQUIRL: **S**oft **Q**-f**u**nctioned Meta-**I**nverse **R**einforcement **L**earning

Test: Single Video Demonstration on a New Task, No Practices

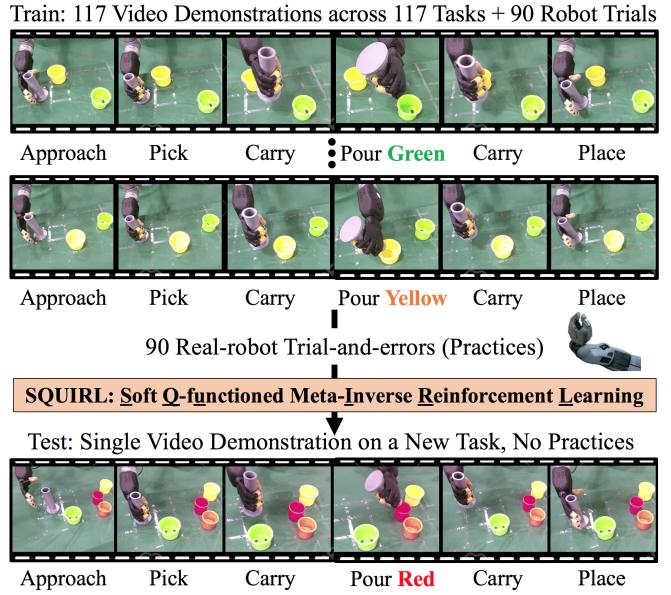| Approach | Pick | Carry | Pour Red | Carry | Place |

Fig. 1: **Learning from a single video demonstration of a long-horizon manipulation task via Soft Q-functioned Meta-IRL (SQUIRL).** In the pick-pour-place example above, the robot needs to approach, pick-up and carry the grey bottle, pour the iron pebble inside the bottle into a specific container, and finally place the bottle back on the table. During training, the robot is given a single video demonstration for *each* of the 117 training tasks. After learning from these 117 videos, the robot also practices 90 trial-and-errors *in total* on these tasks. From such combined expert and robot trajectories, the robot learns the general skills of pouring robustly. At test time, given a single video demonstration of pouring into a *new, unseen* red container at a *new* position, the robot successfully replicates this new task without the need for any trial-and-errors.

hand-designing a proper reward function for every new task. In the case of a vision-based task, video demonstrations also conveniently provide the same pixel state space for the robot.

In learning from demonstrations (LfD), the robot should be sample-efficient in two dimensions – it should use as few expert demonstrations ("demonstrations" hereafter) as possible and take as few trial-and-errors (practices) as possible on its own to learn a robust policy. Among LfD methods, behavioral cloning ("BC" hereafter) is sample-efficient but susceptible to compounding errors. Here, compounding errors refer to the problem in which every time a behavioral-cloned robot makes a small error, it makes a larger error down the road as it drifts away from the expert state distribution. In contrast, IRL alleviates compounding errors by allowing the robot to try the tasks out in the real world and measure its behavior against the expert. However, due to the need to learn a reward function, IRL can require many trial-and-errors in the real world, while BC does not require

Fig.2: **Pick-Pour-Place Robot Setup at Test Time**. Given an RGB image from the top-down (black) *or* $45°$ camera (also black), the UR5-Seed robot is tasked to approach and pick-up the grey cylindrical bottle, pour the iron pebble already inside the bottle into a specific container on the table and finally place the bottle back on the table.

such robot experiences. We posit that leveraging off-policy experiences of trial-and-errors is essential to making IRL sample-efficient enough for real robots. Here, "off-policy experiences" refer to the *cumulative* experiences that the *robot* has collected thus far during *training*. In contrast, "on-policy experiences" are the most recent experiences that the robot has collected using its *current* policy. Humans leverage lifelong, cumulative experiences to learn quickly at present. We envision robots to acquire new skills more quickly by learning from off-policy (i.e., cumulative) experiences.

Finally, many real-world tasks are related and share structures and knowledge that can be exploited to solve a new but similar task later. For example, humans can quickly learn to pick and place a new object after learning to pick and place many known objects. Meta-learning, explicitly utilizing this property, aims to learn a new but related task quickly if it has already learned several similar tasks in the past.

With these motivations, we introduce SQUIRL, a meta-IRL algorithm that learns long-horizon tasks quickly and robustly by learning from 1) video demonstrations, 2) off-policy robot experiences, and 3) a set of related tasks. Fig.1 explains this algorithm using the example of a set of long-horizon pick-pour-place tasks, using the UR5-Seed[1] robot setup shown in Fig.2. In this task, we have the containers (green, yellow, orange, and red), a cylindrical bottle (grey), and an iron pebble inside the bottle. The robot needs to *first approach* and pick-up the grey bottle, pour the iron pebble inside the bottle into a specific container on the table, and then finally place the bottle back on the table, as shown in each row of images in Fig.1. At the beginning of each task, the bottle is *not yet* in hand, but the iron pebble is already in the bottle. At training time, the robot is given a single video demonstration for each of the 117 pick-pour-place tasks, as shown in the first two rows of images in Fig.1. Every new combination of container positions represents a different pick-pour-place task. Furthermore, the robot only needs to pour into *one* of the containers in a single task. Therefore, pouring into different containers also represents different tasks. After learning from these 117 demonstrations, the robot also practices 90 trial-and-errors on these tasks *in total*. From such a combination of expert and robot trajectories, the robot learns the general skills of pick-pour-place robustly. In all 117 training tasks, only *two* of the four containers appear on the table: the green and yellow containers, as shown in

[1]Site: www.seedrobotics.com/rh8d-dexterous-hand.html

the first two rows of images in Fig.1. The orange and red containers are excluded during training and only appear at test time, as shown in the last row of images in Fig.1. We do so to evaluate our algorithm's generalizability to unseen containers *at test time*. As shown in the last row of images in Fig.1, the robot successfully pours into a *new* container (red) at test time, at a *new* position never seen before during training, without the need for any trials or practices.

To achieve such fast generalization to new tasks, our algorithm learns a task encoder network and a task-conditioned policy. The task encoder generates a 32-dimensional task embedding vector that encodes task-specific information. The policy network then learns to generalize to new tasks by accepting this task embedding vector as input, thus becoming "task-conditioned". During training, our algorithm first bootstraps learning by training both the task encoder and the policy jointly via the BC loss. The robot then collects 10 trials across 10 tasks using the warmed-up policy and the task encoder. Next, using the combined experiences of the expert and the robot, our algorithm bypasses reward learning by directly learning a task-conditioned Q-function. Using this Q-function, our algorithm then reuses and re-evaluates all cumulative experiences of the robot to improve the policy quickly. This cycle repeats until the 90th trial. Finally, at test time, the task encoder generates a new task embedding from a *single* video demonstration of a new task. This embedding is then inputted into the task-conditioned policy to solve the new task without any trial-and-errors and yet in a high-performance manner. In summary, our contributions are:

1) A *robust* meta-IRL algorithm that outperforms (90%+ success) its behavioral cloning counterpart in real-robot and simulated vision-based long-horizon manipulation;
2) A *novel* Q-functioned IRL formulation that circumvents reward learning and improves IRL sample efficiency;
3) An *efficient* method that leverages off-policy robot experiences for training and requires no trials at test time;
4) A *general* approach that tackles various long-horizon robotic manipulation tasks and works with both vision and non-vision observations and different action spaces.

II. RELATED WORK

A. Inverse Reinforcement Learning (IRL) and Meta-IRL

Inverse reinforcement learning (IRL) models another agent's (typically the expert's) reward function, given its policy or observed behavior. Previous works have approached the IRL problem with maximum margin methods [1][2] and maximum entropy methods [3][4][5]. In particular, maximum entropy methods recover a distribution of trajectories that have maximum entropy among all distributions and match the demonstrated policy's behaviors. While these methods have shown promising results in continuous control problems, they suffer from low sample efficiency due to the need for evaluating the robot's policy, which can be alleviated by meta-learning (i.e., meta-IRL). SMILe [6] and PEMIRL [7] are two meta-IRL algorithms based on AIRL [8] that leverage a distribution of tasks to learn a continuous task-embedding

space to encode task information and achieve fast adaptation to a new but similar task. Our work differs from [6][7] in four crucial ways. First, our meta-IRL algorithm works with real robots and image observations. Second, instead of a reward function, we directly model a Q-function that the policy can optimize, in order to increase IRL sample efficiency. Third, we train the task encoder with the behavioral cloning (BC) gradient as opposed to the IRL gradient for stabler and more efficient learning. Lastly, we bootstrap policy and task encoder learning using BC before training via meta-IRL.

B. Real-robot Learning from Demonstrations (LfD)

Our work is related to real-robot LfD [9], such as [10][11][12]. In particular, [13] developed IRL on real robots without learning from raw pixels. Other works (e.g., [14][15][16][17]) used BC for real-robot LfD. Another work [18] developed goal-conditioned BC on a simulated robot to learn long-horizon tasks by playing with objects in the scene. While enjoying efficient learning by casting imitation learning into a supervised learning problem, BC suffers from the covariate shift between the train and test data. In comparison, IRL achieves robust performance by modeling the state-action joint distribution instead of the conditional action distribution in BC [19]. Different from previous works, our meta-IRL algorithm works on real-robot *vision-based* tasks, and its Q-functioned IRL policy gradient can be directly combined with the BC gradient signal to approach both the sample efficiency of BC and the robustness of IRL.

C. One-shot Meta-imitation Learning on Real Robots

Our algorithm attempts to enable robots to quickly and robustly imitate a single unseen video demonstration by learning from a distribution of tasks with shared structure, i.e., one-shot robot meta-imitation learning. For example, [20] combines gradient-based meta-learning and BC on a real robot to learn quickly from video demonstrations. [21] then extends [20] to enable robots to learn from human-arm demonstrations directly. [22] then improves [21] to meta-imitation-learn multi-stage real-robot visuomotor tasks in a hierarchical manner. However, constrained by the covariate shift problem of BC, these works show limited task performance (e.g., around 50% success rate for the training tasks). In contrast, our algorithm learns a vision-based manipulation task robustly ($90\%+$ success rates) and efficiently (117 videos, 90 trials) by utilizing the generalization ability of task embeddings [23] and a novel Q-functioned IRL formulation.

III. PRELIMINARIES

A. Off-policy Reinforcement Learning via Soft Actor-Critic

Standard RL models a task $\mathcal{M}$ as an MDP defined by a state space $\mathcal{S}$, an initial state distribution $\rho_0 \in \Pi(\mathcal{S})$, an action space $\mathcal{A}$, a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, a dynamics model $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Pi(\mathcal{S})$, a discount factor $\gamma \in [0, 1)$, and a horizon H. Here, $\Pi(\cdot)$ defines a probability distribution over a set. The robot acts according to stochastic policy $\pi : \mathcal{S} \to \Pi(\mathcal{A})$, which specifies action probabilities for each s. Each policy π has a corresponding $Q^\pi : \mathcal{S} \times \mathcal{A} \to$

$\mathbb{R}$ function that defines the expected discounted cumulative reward for taking an action a from s and following π onward.

Off-policy RL, particularly Soft Actor-Critic (SAC) [24], reuses historical experiences to improve learning sample efficiency by recovering a "soft" Q-function estimator Q_θ. A policy can then be learned by minimizing the KL divergence between the policy distribution and the exponential-Q distribution: $\pi^* = \arg \min_{\pi \in \Pi} D_{KL}(\pi(a \mid s) \parallel \frac{\exp(Q_\theta^{\pi_{old}}(s,a))}{Z(s)})$

B. Timestep-centric IRL as Adversarial Imitation Learning

The purpose of IRL is to learn the energy function f_θ implicit in the provided expert demonstrations and use f_θ to learn a policy that robustly matches the expert performance. In particular, timestep-centric IRL aims to recover an energy function $f_\theta(s, a)$ to rationalize and match the demonstrated expert's action conditional distribution: $p_\theta(a \mid s) = \frac{\exp(f_\theta(s,a))}{Z_\theta} \propto \exp(f_\theta(s,a)) = \overline{p_\theta}(a \mid s)$, where Z_θ is the partition function, an integral over all possible actions given state s. In other words, IRL minimizes the KL divergence between the actual and predicted expert conditional action distributions: $\pi_E(a \mid s)$ and $p_\theta(a \mid s)$.

Adversarial IRL [8][25] provides a sampling-based approximation to MatEntIRL [4] in an adversarial manner. Specifically, AIRL [8] learns a generative policy π_ψ and a binary discriminator D_θ derived from energy function f_θ:

$$D_\theta(s, a) = P((s, a) \text{ is generated by expert})$$
$$= \frac{\overline{p_\theta}(a \mid s)}{\overline{p_\theta}(a \mid s) + \pi_\psi(a \mid s)} = \frac{\exp(f_\theta(s,a))}{\exp(f_\theta(s,a)) + \pi_\psi(a \mid s)} \quad (1)$$

and θ is trained to distinguish state-action pairs sampled from the expert vs. the policy, using binary cross entropy loss:

$$\mathcal{L}^{IRL} = -\mathbb{E}_{(s,a) \sim \pi_\psi, \pi_E}[y(s, a) \log(D_\theta(s, a)) + (1 - y(s, a)) \log(1 - D_\theta(s, a))] \quad (2)$$

where $y(s, a) = \mathbb{1}\{(s, a) \text{ is generated by expert } \pi_E\}$.

Meanwhile, the policy is trained to maximize the MaxEntIRL Objective [4], or equivalently, to match the expert's state-action joint distribution via reverse KL divergence [19].

C. One-shot Meta-imitation Learning from A Single Video

In one-shot meta-imitation learning, the robot is trained to solve a large number of tasks drawn from a task distribution $p(\mathcal{M})$. The total number of tasks in this task distribution can be finite or infinite. Each imitation task $\mathcal{M}_{train}^i$ consists of a single video demonstration $\mathcal{D}_{\pi_E}^i$. During training, the robot can also generate limited practice trajectories (e.g., 90). For example, in the Pick-Pour-Place experiment in Fig.1, the robot receives a single video demonstration for each of the 117 tasks. Each task is characterized by a different combination of container positions, or pouring into the green vs. the yellow container. At test time, the robot receives a single video of a new task $\mathcal{M}_{test}^i$ drawn from $p(\mathcal{M})$. For example, a new Pick-Pour-Place test task can be a *new* combination of container positions or pouring into a *new* container (e.g., the red or orange container). The robot then needs to solve this task the first time *without trial-and-error*.

D. Embedding-based Meta-learning

Embedding-based meta-learning [7][23] learns a task-specific embedding vector z that contains task-level abstraction to adapt to a new but related task quickly. This method aims to learn a task-conditioned policy $\pi(a|s, z)$ that maximizes task-conditioned expected returns: $\max_\pi \mathbb{E}_{(s_t, a_t) \sim \pi, \rho_0} [\sum_{t=1}^T r(s_t, a_t|c) + \alpha\mathcal{H}(\pi(a_t|s_t, c))]$, by learning an embedding space Z that maximizes the mutual information between z and task context c. The goal is to make this learned embedding space generalizable to new tasks so that at test time, the policy can quickly adapt to unseen tasks with no or few practices. A key advantage of embedding-based meta-learning is the ability to learn from off-policy experiences. However, current methods are mostly if not only demonstrated in non-vision tasks in simulation.

IV. MATHEMATICAL FORMULATION FOR SQUIRL

A. SQUIRL: Timestep-centric IRL as Soft Q-Learning

Previous works in timestep-centric IRL such as [6][7][8] have interpreted the energy function f_θ in Eq.1 as a reward function r_θ and later recover a Q or advantage function based on reward r_θ for policy improvement. To improve IRL sample efficiency, we propose to *bypass* this reward learning and directly interpret $f_\theta(s, a)$ as the soft Q-function [24] $Q_\theta^{\pi_{mix}}(s, a)$. This soft Q-function models the expert's behavior as maximizing both the Q-value and its *entropy* (i.e., randomness) simultaneously. It also encourages the robot to explore the real world to imitate the expert more robustly. Under this formulation, approximating the expert's conditional action distribution is equivalent to recovering a soft Q-function under which the expert is soft Q-optimal:

$$\arg\min_\theta D_{KL}(\pi_E(a \mid s) \parallel p_\theta(a \mid s))$$
$$= \arg\max_\theta \mathbb{E}_{a \sim \pi_E(a|s)}[Q_\theta^{\pi_{mix}}(s, a)] - \log Z_\theta \quad (3)$$

Eq.3 rationalizes the expert behavior intuitively because the expert should be optimal with respect to the *cumulative* reward [3], not the immediate reward. Here, $Q_\theta^{\pi_{mix}}$ is under a mixture policy π_{mix} between the robot and expert's policies.

B. SQUIRL as Expert Imitation and Adversarial Learning

Under SQUIRL, the policy learning objective (Eq.4) is also equivalent (derivations on website) to matching: 1) the exponential-Q distribution of the discriminator θ (Eq.5), 2) the generator's objective in Generative Adversarial Networks (GANs) [26] (Eq.6), and 3) the joint state-action distribution of expert [19] (Eq.7): $\pi^* = \arg\min_{\pi \in \Pi} \mathcal{L}^{RL}(\pi)$, where

$$\mathcal{L}^{RL}(\pi) = D_{KL}(\pi_\psi(a \mid s) \parallel \frac{\exp Q_\theta^{\pi_{mix}}(s, a)}{Z(s)}) \quad (4)$$

$$= D_{KL}(\pi_\psi(a \mid s) \parallel p_\theta(a \mid s)) \quad (5)$$

$$= \mathbb{E}_{(s, a) \sim \pi_{mix}}[\log(1 - D_\theta(s, a)) - \log(D_\theta(s, a))] \quad (6)$$

$$= D_{KL}(\rho_{\pi_\psi}(s, a) \parallel \rho_{\pi_E}(s, a)) \quad (7)$$

Meanwhile, the discriminator θ is matching its Q-function to the log-distribution of the expert's conditional action distribution (Section III-B). Therefore, when this Q-function is

optimal: $Q_\theta^{\pi_{mix}} = Q_{\theta*}^{\pi_{mix}}$, the robot's policy objective (Eq.4) is also matching the expert's conditional action distribution:

$$\psi^* = \arg\min_\psi E_{\rho_{\pi_{mix}}(s)}[D_{KL}(\pi_\psi(a \mid s) \parallel \pi_E(a \mid s))] \quad (8)$$

C. Comparison to the Behavioral Cloning (BC) Objective

While BC attempts to learn a policy that also matches the expert's conditional action distribution [19], the *fundamental* difference is that the KL-divergence in BC's case is computed under the *expert's* narrow state distribution $\rho_{\pi_E}(s)$: $\psi_{BC}^* = \arg\min_\psi E_{\rho_{\pi_E}(s)}[D_{KL}(\pi_E(a \mid s) \parallel \pi_\psi(a \mid s))]$. In contrast, ours (Eq.8) is computed under $\rho_{\pi_{mix}}(s)$: the state distribution of the *combined cumulative* experience of the robot and the expert, which is a much wider distribution than the expert distribution. We hypothesize that this, along with matching the joint state-action distribution of the expert (Eq.7), makes our algorithm less susceptible to compounding errors than BC, as experimentally tested in Section VI.

V. SQUIRL: SOFT Q-FUNCTIONED META-IRL

Shown in Fig.3, our algorithm learns three neural networks jointly – a task encoder (yellow), a task-conditioned policy (orange), and a task-conditioned soft Q-function (green):

1) $\Psi_\phi(c)$: a **task encoder** that encodes a sampled batch of $C = 64$ expert state-action pairs $c = \{s_{1:C}^i, a_{1:C}^i\}$ from a task i into a single 32-dim embedding vector $z^i \in \mathbb{R}^{32}$ (by computing the mean vector across 64 embeddings) that enables generalization to new tasks. This batch of expert state-action pairs is randomly sampled and thus does not encode time information. Both the policy and the Q-function accept this embedding vector as input.

2) $\pi_\psi(s, z^i)$: a **task-conditioned policy** the robot uses to perform a task i given state s and the task embedding vector $z^i \in \mathbb{R}^{32}$ outputted by the task encoder $\Psi_\phi(c)$.

3) $Q_\theta(s, a, z^i)$: a **task-conditioned soft Q-function** used to train the policy $\pi_\psi(s, z^i)$ to more robustly mimic the expert's behavior for the robotic manipulation task i.

To begin, the robot is given an expert trajectory of state-action pairs $\mathcal{D}_{\pi_E}$ for each of the 117 training tasks. The robot first uses these expert trajectories to bootstrap training for both its policy π_ψ, and the task encoder Ψ_ϕ via behavioral cloning (Eq.9). This way, the robot can distinguish the train tasks better and learn more quickly in the real world. Next, the robot generates 10 trials (state-action pairs) $\overline{\mathcal{D}}_{\pi_\psi}$ *in the physical world* (not simulation) using its warmed-up policy and task encoder. Then, the robot uses both the expert's and its state-action pairs to train a discriminator θ. This discriminator classifies which state-action pairs come from the expert π_E vs. the robot π_ψ. At first, the robot is distinctly worse than the expert at performing the tasks. This makes it easy for the discriminator to classify. By doing so, the discriminator learns a Q-function $Q_\theta^{\pi_{mix}}$ using Eq.3.

Using the learned Q-function $Q_\theta^{\pi_{mix}}$, the robot trains its policy π_ψ via Eq.4. Meanwhile, the robot also has the option to continue updating its task-conditioned policy and task encoder via behavioral cloning (Eq.9). Since training the policy via Eq.4 is equivalent to indirectly imitating the expert

Step 1: Warm-up Policy and Task Encoder via BC

117 Expert Demonstrations across 117 tasks: $s_{1:T}, a_{1:T}$

Behavioral Cloning

Policy $\pi_\psi(a \mid s, z)$

Task Encoder $\Psi_\phi(c)$

Step 2: Improve Policy via SQUIRL

Collect *one* Real-Robot Trial per Task i

Add to Cumulative Dataset

Repeat until Convergence (90 total trials)

Sample $m = 10$ tasks $i_{1:m}$ and m embeddings $z^{i_{1:m}}$

Improve Q Function $Q_\theta(s, a, z^i)$

Improve Policy $\pi_\psi(a \mid s, z)$

Train

Solve New Task with No Trials using Policy and Task Encoder

Policy $\pi_\psi(a \mid s, z)$

Task Embedding z

Task Encoder $\Psi_\phi(c)$

A New Single Expert Demonstration $s_{1:T}, a_{1:T}$
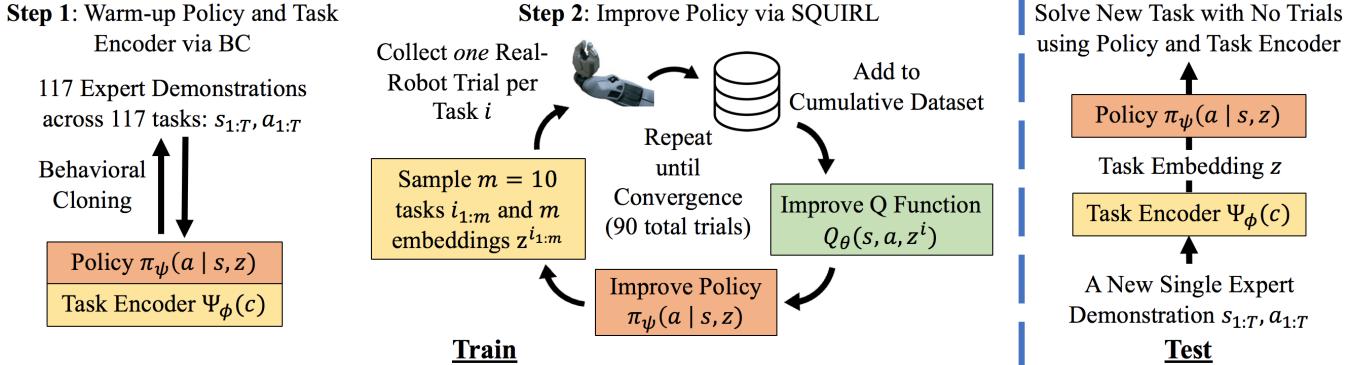
Test

Fig. 3: **SQUIRL: Soft Q-functioned Meta-IRL**. To begin, our algorithm bootstraps learning for the policy (orange) and the task encoder (yellow) via behavioral cloning (the left third of Fig.3). Next, our algorithm uses the warmed-up policy and task encoder to generate 10 trials *in the physical world* (not in simulation). Using the combined expert and robot trajectories, our algorithm learns a task-conditioned soft Q-function (green) that rationalizes the expert's behaviors as maximizing both cumulative reward and entropy (i.e., randomness). Using this Q-function, our algorithm then quickly improves the policy using *all cumulative* robot and expert timesteps. This cycle repeats until convergence, totaling *90 trials* (the middle third of Fig.3). Finally, at test time (the right third Fig.3), our algorithm generates a new embedding z for the new task, and inputs this embedding into the task-conditioned policy to solve the new task without any practices.

(Eq.7 and 8), as derived in Section IV-B, the trajectories generated by the policy gradually become more similar to the expert. This makes the state-action pairs more difficult for the discriminator to classify. This difficulty, in turn, forces the discriminator to learn a more precise Q-function, which then encourages the policy to mimic the expert even more closely. This cycle repeats until convergence (90 trials *in total*), at which point: 1) the policy matches the expert performance, 2) the task encoder learns to generalize to new tasks, and 3) the discriminator continues to struggle to distinguish state-action pairs correctly despite having learned an accurate Q-function.

A. Rationale for Bypassing Reward Learning via SQUIRL

SQUIRL learns a Q-function without rewards because 1) the policy is ultimately trained by the Q-function, not rewards, thus bypassing reward learning improves IRL sample efficiency, and 2) circumventing reward learning avoids off-policy Q-learning from a *constantly changing* reward function and makes training easier and more stable empirically.

B. Architectures for Policy, Task Encoder, and Q-function

For all non-vision tasks, we parameterize $\pi_\psi, \Psi_\phi, Q_\theta$ with five fully-connected (FC) layers. For vision tasks, we use a 5-layer CNN followed by a spatial-softmax activation layer for the RGB image. This activation vector is then concatenated with the non-vision input vector and together passed through five FC layers. Our algorithm is *general* and works with many other network architectures, state, and action spaces.

C. Incorporating BC to Bootstrap and Accelerate Learning

Since our algorithm's IRL objective (Eq.8) is compatible with BC, as explained in Section IV-C, our algorithm can jointly be trained with BC to stabilize and accelerate learning without conflicting gradient issues (line 16 in Algorithm 1):

$$\mathcal{L}^{BC} = \mathbb{E}_{(s,a)\sim\pi_E}[\|\pi_\psi(s, \Psi_\phi(c)) - a\|^2] \quad (9)$$

This, combined with the off-policy nature of our algorithm, also allows the robot to bootstrap learning by first "pre-training" via BC (Eq.9) using the expert demonstrations, before improving performance further via meta-IRL training.

Algorithm 1 SQUIRL: Soft Q-functioned Meta-IRL (Train)

Input: One expert video demonstration trajectory of state-action pairs $\mathcal{D}_{\pi_E}^i = \{s_{1:H}^i, a_{1:H}^i\}$ for each of the n training tasks $i = 1 : n$, where H is the horizon of the task (e.g., $n = 117, H = 100$)

1: Initialize soft Q-function Q_θ, policy π_ψ, task encoder Ψ_ϕ, and an empty buffer of off-policy robot trajectories $\mathcal{D}_{\pi_\psi}^i \leftarrow \{\}$ for each training task $i = 1 : n$
2: Warm-up policy and task encoder via $\mathcal{L}^{BC}$ (Eq.9)
3: **while** not converged **do**
4:     Sample a batch of m task indices $\{i^{1:m}\}$ from all training tasks $i = 1 : n$, (e.g., $m = 10$)
5:     **for** $i = i^{1:m}$ **do**
6:         Infer task embedding $z^i = \mathbb{R}^{\mathcal{Z}} \leftarrow \Psi_\phi(c)$, where $c = \{s_{1:C}^i, a_{1:C}^i\} \sim \mathcal{D}_{\pi_E}^i$ (e.g., $\mathcal{Z} = 32, C = 64$)
7:         Generate a robot trajectory of state-action pairs $\overline{\mathcal{D}}_{\pi_\psi}^i = \{s_{1:H}^i, a_{1:H}^i\}$ from task i using π_ψ, z^i
8:         $\mathcal{D}_{\pi_\psi}^i \leftarrow \mathcal{D}_{\pi_\psi}^i \cup \overline{\mathcal{D}}_{\pi_\psi}^i$
9:     **end for**
10:     **for** $j = 1 : J$ (e.g., $J = 400$) **do**
11:         Sample another batch of m task indices $\{i^{1:m}\}$
12:         $\theta \leftarrow \theta - \nabla_\theta \mathcal{L}^{IRL}$ (Eq.2) using a combined batch of $\mathcal{B} = 128$ robot and expert timesteps: $\overline{\mathcal{D}}_{\pi_\psi}^i \cup \overline{\mathcal{D}}_{\pi_E}^i$ and z^i, where $\overline{\mathcal{D}}_{\pi_\psi}^i \sim \mathcal{D}_{\pi_\psi}^i, \overline{\mathcal{D}}_{\pi_E}^i \sim \mathcal{D}_{\pi_E}^i, i = \{i^{1:m}\}$
13:     **end for**
14:     **for** $k = 1 : K$ (e.g., $K = 2000$) **do**
15:         Sample another batch of m task indices $\{i^{1:m}\}$
16:         **if** necessary **then** $\{\psi, \phi\} \leftarrow \{\psi, \phi\} - \nabla_{\psi,\phi}\mathcal{L}^{BC}$ (Eq.9) using a batch of $\mathcal{B}$ expert timesteps $\overline{\mathcal{D}}_{\pi_E}^i \sim \mathcal{D}_{\pi_E}^i, z^i, i = \{i^{1:m}\}$ **end if**
17:         $\psi \leftarrow \psi - \nabla_\psi \mathcal{L}^{RL}$ (Eq.4) using a combined batch of $\mathcal{B}$ robot and expert timesteps: $\overline{\mathcal{D}}_{\pi_\psi}^i \cup \overline{\mathcal{D}}_{\pi_E}^i$ and z^i, where $\overline{\mathcal{D}}_{\pi_\psi}^i \sim \mathcal{D}_{\pi_\psi}^i, \overline{\mathcal{D}}_{\pi_E}^i \sim \mathcal{D}_{\pi_E}^i, i = \{i^{1:m}\}$
18:     **end for**
19: **end while**
20: **return** soft Q-function Q_θ, policy π_ψ, task encoder Ψ_ϕ

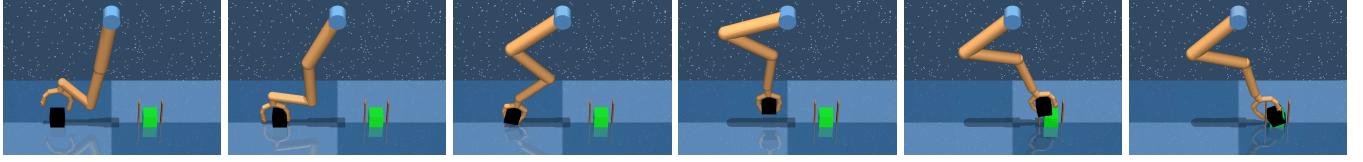| Approach Box | Lower to Box | Grasp Box | Pick up Box | Carry Box | Drop Box |

Fig. 4: **Pick-Carry-Drop Experiment.** The robot needs to approach, lower to, grasp, pick-up, carry, and drop the box to solve the task.

Algorithm 2 SQUIRL: Soft Q-functioned Meta-IRL (Test)

Input: π_ψ, Ψ_ϕ, Q_θ, and a single expert video demonstration of state-action pairs $\mathcal{D}^i_{\pi_E} = \{s^i_{1:H}, a^i_{1:H}\}$ from a *new* task i unseen during training

1: Infer task embedding vector $z^i = \mathbb{R}^\mathcal{Z} \leftarrow \Psi_\phi(c)$, where $c = \{s^i_{1:C}, a^i_{1:C}\} \sim \mathcal{D}^i_{\pi_E}$ (e.g., $\mathcal{Z} = 32, C = 64$)
2: Rollout robot trajectory in the real world using π_ψ, z^i

D. Using Expert Demonstration as Both the Input Task Context Variables and Training Signal for the Task Encoder

Learning robust task embeddings enables robots to generalize to new tasks quickly [23]. To this end, our algorithm proposes to use 64 expert timesteps as the input task context variable c into the task encoder, as opposed to 64 robot timesteps. This is because context variables should explore the task and environment sufficiently well to expose the key information of the task, and expert demonstration timesteps are an ideal candidate compared to the timesteps from the robot's suboptimal policy. As a result, the context variable c input into the task encoder only includes the states and actions of the expert, but not the rewards or the next states.

In addition, we choose the BC loss $\mathcal{L}^{BC}$ in Eq.9 as the training loss for learning the task encoder Ψ_ϕ. This BC loss is stable since the expert timesteps are fixed. In contrast, the IRL loss $\mathcal{L}^{IRL}$ (Eq.2) and the policy loss $\mathcal{L}^{RL}$ (Eq.4) are less stable because the training data distribution for both losses are non-stationary. This design choice also allows us to learn a robust task embeddings first via BC pre-training before performing meta-IRL training via SQUIRL. We empirically observe that such pre-training can improve the training stability and the sample efficiency of SQUIRL, but the final policy performance is similar with or without BC pre-training. In summary, our algorithm is detailed in Algorithm 1 (train) and Algorithm 2 (test), with hyperparameters detailed here[2].

VI. EXPERIMENTS AND RESULTS ANALYSIS

We evaluate the generality and robustness of our algorithm across long-horizon vision and non-vision tasks with continuous state and action spaces in both simulation (Pick-Carry-Drop, a horizon of 1024 timesteps, 30 train tasks) and real-world (Pick-Pour-Place, a horizon of 100 timesteps, 117 train tasks). There is only a *single* expert demonstration for *each* of the train or test tasks. We compare with the PEARL-BC baseline, which is the behavioral cloning version of PEARL [23].

[2]Hyperparameters in Algorithm 1 and 2. Policy gradient batch size $\mathcal{B}$: 1024 (non-vision), 128 (vision); task embedding batch size C: 64; all learning rates: $3e^{-4}$; starting SAC alpha: $1e^{-5}$; SAC target entropy: -300; IRL updates per epoch J: 400; policy updates per epoch K: 2000; task embedding size $\mathcal{Z}$: 32; meta-batch size m: 10; discount rate γ: 0.99

Evaluation: We evaluate real-robot and simulation experiments on **50** and **500** trials respectively across **50** seen and unseen tasks. We report mean and standard deviation ("stdev" hereafter). The performance difference between different experiments is statistically significant if the difference in **mean** is at least **either** standard deviation away. Experimental video is at http://crlab.cs.columbia.edu/squirl.

A. Simulation Experiment: Pick-Carry-Drop

Description. We modify the planar Stacker task [27] to create "Pick-Carry-Drop". Shown in Fig.4, a robot is tasked to approach, pick, carry, and drop the black box into the stack marked in green. The task is successful if the box is dropped into the stack within 1024 timesteps, and failed otherwise.

State Space. We evaluate our algorithm on both the vision and the non-vision version of the task, to demonstrate that SQUIRL is *general* across different state space modalities. The state space for the vision version includes 1) the joint angles and velocities for its 5-DOFs, 2) a one-hot vector indicating the current stage of the task, and 3) an RGB image shown in Fig.4. The non-vision version's state space replaces the RGB image with the *position of the black box*.

Action Space. The robot controls its 5-DOF *joint torques*.

Task Definition. There are a total of 30 training tasks in this experiment, each corresponding to a different *drop location*: $x \in \{-0.15, -0.14, \dots, 0.14\}$. During test time, we randomly sample a new, *real-valued* drop location from the maximum valid range: $x \in [-0.25, 0.25]$. The green drop location is *invisible* in both the vision and the non-vision version of the task. Therefore, the robot needs to infer the green drop location (i.e., task information) solely from the provided expert video demonstration. On the other hand, the starting pose of the robot and the location of the black box are all initialized randomly at the beginning of each task.
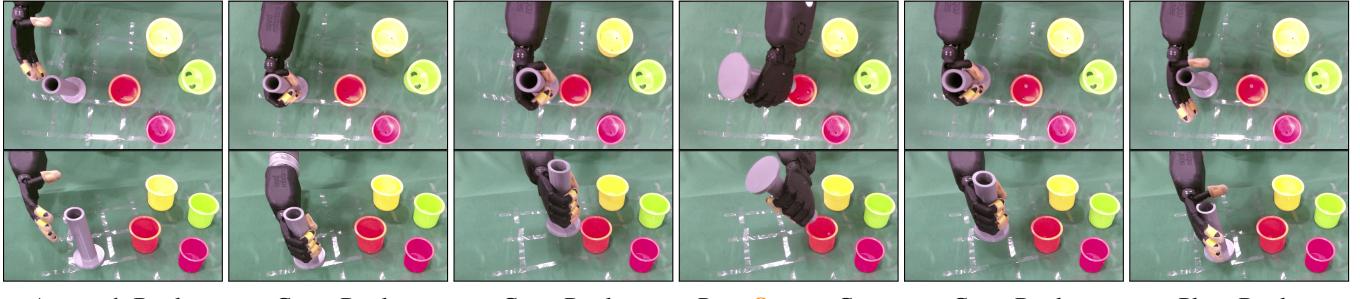
Robot Trials. The robot uses 150 training trials *in total*.

Expert Demonstration. We trained an expert policy from scratch via RL to provide expert demonstrations. The reward function used to train the expert policy comprises of six stages, each stage with a reward of 10. Designing this reward function has taken significant human effort, which exhibits the value of directly learning from video demonstrations.

TABLE I: Pick-Carry-Drop Results (% Drop Success±Stdev)

Tasks	Seen	Unseen	Seen	Unseen
	Vision		Non-Vision	
SQUIRL (BC + IRL)	**95.8**±**1.7**	**95.0**±**1.5**	**97.3**±**3.0**	**96.9**±**2.0**
Baseline (PEARL-BC)	77.8±1.6	76.5±0.7	90.8±2.5	89.5±1.6
Ablation: No BC Joint Training or BC Pre-training				
SQUIRL (IRL Only)	93.8±1.8	93.2±1.6	94.7±1.7	93.9±1.4

Simulation Results and Analysis. As shown in Table I, our algorithm, "SQUIRL (BC + IRL)", pre-trains via BC and

| Approach Bottle | Grasp Bottle | Carry Bottle | Pour Orange Cup | Carry Bottle | Place Bottle |

Fig. 5: **Pick-Pour-Place at Test Time.** To solve this task, the robot needs to first approach, grasp and carry the grey bottle, pour the iron pebble inside the bottle into a specific container, and carry and place the bottle back on the table. At the beginning of each task, the bottle is not in hand, and but the iron pebble is already in the bottle. Top row: top-down camera images. Bottom row: 45°camera images.

then trains the policy using both the BC loss (Eq.9) and the IRL policy gradient loss (Eq.4). It statistically significantly outperforms the PEARL-BC baseline in both the vision (95.8%±1.7 vs. 77.8%±1.6) and non-vision (97.3%±3.0 vs. 90.8%±2.5) version of the task for seen tasks. For unseen tasks, we observed similar outperformance (95.0%±1.5 vs. 76.5%±0.7 in the vision case and 96.9%±2.0 vs. 89.5%±1.6 in the non-vision case). Qualitatively, in the PEARL-BC's case, the robot sometimes misses the drop location as it attempts to drop the box or fails to pick up the box when the box gets stuck by the walls of the stack (kindly see website). The performance drop of the baseline from the non-vision version (90.8%±2.5 and 89.5%±1.6 for seen and unseen tasks) to the vision version (77.8%±1.6 and 76.5%±0.7 for seen and unseen tasks) is mainly because vision-based manipulation tends to suffer from larger compounding errors. Nevertheless, as evident in the statistical similarities between seen and unseen tasks for SQUIRL (95.8%±1.7 vs. 95.0%±1.5 for vision) and PEARL-BC (77.8%±1.6 vs. 76.5%±0.7 for vision), both algorithms can generalize to unseen tasks, due to the generalizability of task embeddings.

Ablation: IRL Gradient Only. To compare the performance contribution of SQUIRL's meta-IRL core training procedure directly against PEARL-BC, we created "SQUIRL (IRL only)", which trains the policy using only the policy gradient loss in Eq.4 (no BC joint training or pre-training). This *ablated* version still outperforms the PEARL-BC baseline (93.8%±1.8 vs. 77.8%±1.6 for seen vision tasks, 93.2%±1.6 vs. 76.5%±0.7 for unseen vision tasks). Nevertheless, by combining BC and IRL gradients, "SQUIRL (BC + IRL)" improves performance slightly further (95.8%±1.7 and 95.0%±1.5). Intuitively, while BC only matches the expert's conditional action distribution under the *expert's* state distribution, BC's supervised learning signal is stabler than IRL. Joint training with BC and IRL gradients can be interpreted as combining the stability of BC and the robustness of Q-functioned IRL, by matching the conditional action distribution of the expert under the broader state distribution of the expert-robot mixture experience (Eq.8), in addition to matching the expert's joint state-action distribution (Eq.7).

B. Real-Robot Experiment: Pick-Pour-Place

Description. We evaluated our algorithm on the UR5-Seed robot (Fig.2) to perform a set of long-horizon pick-pour-place tasks. As shown in Fig.2, in each task, there is a grey cylindrical bottle, an iron pebble that is already in the bottle, and more than one container on the table. The robot is tasked to approach and pick-up the grey bottle, pour the iron pebble into a specific container, and place the bottle back on the table. The task is a success only if the pebble is poured into the *correct* container and the bottle is placed upright on the table within $H = 100$ timesteps, and a failure otherwise.

State Space. The state space contains a top-down or 45°camera's RGB image (Fig.5), and 2 binary indicators for whether the robot has poured or closed the hand, respectively.

Action Space. The action space includes the Cartesian unit directional vector for the end-effector movement. During each timestep, the robot can adjust the end-effector by 2cm along any 3D direction. The action space also includes a binary indicator to control the arm vs. the hand and a trinary indicator to close, open, or rotate the hand for pouring.

Orthogonality to State and Action Representations. While Pick-Pour-Place can be tackled by first localizing the correct container via object detection (alternative state space) and then executing motion-planning trajectories to pour (alternative action space), our algorithm is *general* across and orthogonal to alternative state and action spaces.

Task Definition. As shown in each row of images in Fig.1, each task is defined by the positions and colors of the containers, and by the correct container to pour into. There are *always only* the green and yellow containers in the 117 train tasks. 25 of the 50 test tasks have the green and yellow containers at *new* positions. The remaining 25 test tasks *add* the red and the orange *unseen* containers, or either. Since there is always more than one container in the RGB image, the robot will not know which container to pour into *without* the expert demonstration. Therefore, the robot needs to depend solely on the task encoder's ability to extract the correct task information from the expert demonstration.

Robot Trials. The robot collects 90 training trials *in total*.

Expert Demonstration. We collect demonstrations via teleoperation using a Flock of Birds sensor[3]. Using the human wrist pose detected by the sensor in real-time, we move, open, close, or rotate the robot hand for pouring. We collected 117 video demonstrations across 117 tasks for training. It takes 1-2 minutes to collect one demonstration.

---

[3]Flock of Birds is a 6D pose tracker from Ascension Technologies Corp.

TABLE II: Pick-Pour-Place Results (% Pour Success±Stdev)

Tasks	RGB Image	Seen	Unseen
SQUIRL (BC + IRL)		**92.0±4.5**	**90.0±7.1**
Baseline (PEARL-BC)	Top-Down (90°)	70.0±7.1	68.0±11.0
Baseline (Standard-BC)		60.0±10.0	56.0±11.4
SQUIRL (BC + IRL)	45° (**Ablation**)	90.0±7.1	88.0±8.4

Real-robot Results and Analysis. As shown in Table II, our algorithm outperforms the PEARL-BC baseline statistically significantly in both seen tasks (92.0%±4.5 vs. 70.0%±7.1) and unseen tasks (90.0%±7.1 vs. 68.0%±11.0). This observed outperformance mainly originates from our soft Q-functioned IRL formulation, which forces the robot to imitate the expert under a much wider state distribution provided by the expert-robot mixture trajectories, instead of the narrow state distribution of the expert demonstrations. This helps reduce compounding errors during task execution. The low performance of the PEARL-BC baseline is mainly due to additional compounding errors induced by real-world sensory noises such as unstable lighting conditions and small perturbation to camera positions. Qualitatively, the PEARL-BC baseline sometimes pours into the wrong container, misses the target container by a few centimeters, or moves past the target container while failing to pour in time (kindly see website for examples). Nevertheless, from the statistical similarity between seen and unseen tasks for both our algorithm (92.0%±4.5 vs. 90.0%±7.1) and PEARL-BC (70.0%±7.1 vs. 68.0%±11.0), we see that the learned task encoder is still effectively generalizing to a new, related task.

Comparison to the "Standard-BC" Baseline. We also compared to "Standard-BC" (60.0%±10.0 and 56.0%±11.4 for seen and unseen tasks), which performs no meta-learning and learns every train or test task *independently* from scratch via BC. As a result, the neural network *overfits* to the single demonstration and fails to generalize to real-world sensory (camera) noises at test time. Note that Standard-BC's unseen-task performance is slightly lower than seen tasks since the unseen tasks are more challenging with at most 4 containers on the table, compared to only 2 containers in seen tasks.

Ablation: Non-top-down Camera. We also tested our algorithm with a 45° RGB image (90.0%±7.1 and 88.0%±8.4 for seen and unseen tasks) against a top-down RGB image (92.0%±4.5 and 90.0%±7.1 for seen and unseen tasks). The statistical similarity between the two shows that SQUIRL is *general* and can accept a non-top-down RGB input image.

VII. CONCLUSION

We introduced SQUIRL, a robust, efficient, and general Soft Q-functioned meta-IRL algorithm, towards enabling robots to learn from limited expert (one per task) and robot (90 in total) trajectories. This algorithm is statistically significantly more robust than behavioral cloning and requires no trial-and-errors at test time. Finally, this general algorithm has been tested to work with various long-horizon manipulation tasks, and across vision and non-vision state and action spaces. In the future, we will extend this algorithm to learn from direct human-arm demonstrations instead of teleoperation. This will lower the cost of collecting real-world expert demonstrations further. We also aim to incorporate hierarchical learning into SQUIRL to solve much longer horizon manipulation tasks by reusing low-level subpolicies.

REFERENCES

[1] P. Abbeel and A. Ng, "Apprenticeship learning via inverse reinforcement learning," *International Conference on Machine Learning*, 2004.

[2] N. Ratliff, B. Andrew, and M. Zinkevich, "Maximum margin planning," *International Conference on Machine Learning (ICML)*, 2006.

[3] B. Ziebart, "Modeling purposeful adaptive behavior with the principle of maximum causal entropy," *PhD Thesis*, 2010.

[4] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proc. AAAI*, 2008.

[5] A. Boularias, J. Kober, and J. Peters, "Relative entropy inverse reinforcement learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 15. PMLR, 11–13 Apr 2011.

[6] S. K. Seyed Ghasemipour, S. S. Gu, and R. Zemel, "Smile: Scalable meta inverse reinforcement learning through context-conditional policies," in *Advances in Neural Information Processing Systems*, 2019.

[7] L. Yu, T. Yu, C. Finn, and S. Ermon, "Meta-inverse reinforcement learning with probabilistic context variables," in *NeurIPS*, 2019.

[8] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adverserial inverse reinforcement learning," in *ICLR*, 2018.

[9] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[10] D. Xu, S. Nair, Y. Zhu, J. Gao, A. Garg, L. Fei-Fei, and S. Savarese, "Neural task programming: Learning to generalize across hierarchical tasks," in *International Conference on Robotics and Automation*, 2018.

[11] D.-A. Huang, S. Nair, D. Xu, Y. Zhu, A. Garg, L. Fei-Fei, S. Savarese, and J. C. Niebles, "Neural task graphs: Generalizing to unseen tasks from a single video demonstration," in *CVPR*, 2019.

[12] D.-A. Huang, Y.-W. Chao, C. Paxton, X. Deng, L. Fei-Fei, J. C. Niebles, A. Garg, and D. Fox, "Motion reasoning for goal-based imitation learning," in *ICRA*, 2020.

[13] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *ICML*, 2016.

[14] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *ICRA*, 2018.

[15] J. Kober and J. Peters, "Imitation and reinforcement learning - practical algorithms for motor primitive learning in robotics," *IEEE Robotics and Automation Magazine*, vol. 17, no. 2, pp. 55–62, 2010.

[16] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *International Conference on Robotics and Automation (ICRA)*, 2009.

[17] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain, "Time-contrastive networks: Self-supervised learning from video," in *ICRA*, 2018.

[18] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning latent plans from play," in *CoRL*, 2019.

[19] S. K. S. Ghasemipour, R. Zemel, and S. Gu, "A divergence minimization perspective on imitation learning methods," in *CoRL*, 2019.

[20] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," in *CoRL*, 2017.

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

[22] T. Yu, P. Abbeel, S. Levine, and C. Finn, "One-shot hierarchical imitation learning of compound visuomotor tasks," in *IROS*, 2019.

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

[24] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *International Conference on Machine Learning (ICML)*, 2018.

[25] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in neural information processing systems*, 2016.

[26] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014.

[27] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. de Las Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. Lillicrap, and M. Riedmiller, "DeepMind control suite," Tech. Rep., Jan. 2018.