Don’t Start From Scratch: Leveraging Prior Data to Automate Robotic Reinforcement Learning

Author Names Omitted for Anonymous Review. Paper-ID 222

***Abstract*—Reinforcement learning (RL) algorithms typically require a substantial amount of data, which may be time-consuming to collect with a robot, as well as the ability to freely return to an initial state to continue practicing a task, which requires laborious human intervention in the real world. Moreover, robotic policies learned with RL often fail when deployed beyond the carefully controlled setting in which they were learned. In this work, we demonstrate that these varied challenges of real-world robotic learning can all be tackled by effective utilization of diverse offline interaction datasets collected from previously seen tasks. While much prior work on robotic RL has focused on learning from scratch, and has attempted to solve each of the above problems in isolation, we devise a system that uses prior offline datasets to tackle all of these challenges together. Our system first uses techniques from offline reinforcement learning to extract useful skills and representations from prior offline data, which gives the agent a baseline ability to perceive and manipulate the world around it. Then, when faced with a new task, our system adapts these skills to quickly learn to both perform the new task *and* return the environment to an initial state, effectively learning to perform its own environment reset. We show that training on prior data gives rise to behaviors that generalize to far more varied conditions, than simply not using this data. We evaluate our method on a suite of challenging robotic manipulation tasks, involving high-dimensional visual observations and sparse binary reward functions, both in the real world and in simulation. Our empirical results demonstrate that incorporating prior data into robotic reinforcement learning enables autonomous learning, substantially improves sample-efficiency of learning, and results in policies that generalize better.**

I. Introduction

Reinforcement learning (RL) provides a concise and appealing abstraction that captures the problem of learning behavioral skills for embodied systems such as robots: by framing the problem as experiential utility maximization, RL provides a general learning-based framework that, in principle, could be utilized to acquire any goal-directed behavior. However, in practice, the standard RL problem formulation overlooks many of the challenges that arise in real-world robotic learning. RL problems are classically (though not exclusively) situated in settings where ample exploration can be performed from scratch, the environment can be reset episodically, and the focus is more on attaining the highest possible performance rather than good generalization, as for example in the case of playing a board game or video game. However, real-world robotic learning problems have very different constraints. With real-world robots, online interaction and exploration is often at a premium, the robot must figure out how to reset the environment itself between attempts, and the natural variability and uncertainty of the real world makes generalization far more essential than squeezing out every bit of final performance.

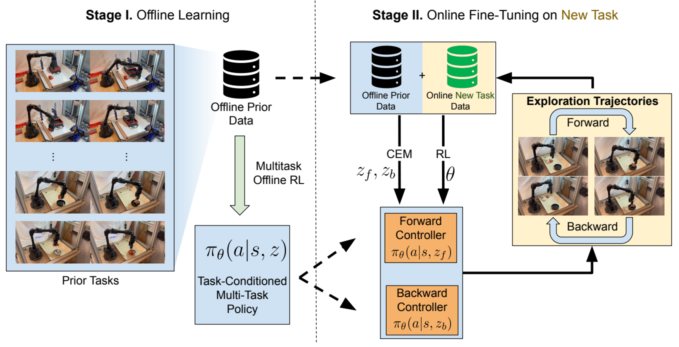

Fig. 1. **Overview of our system.** In the offline stage (left), we train a multi-task policy that captures prior knowledge from an offline dataset of previously experienced tasks. Then, in the online stage (right), this multi-task policy is used to initialize learning for a new task, providing both a forward policy and a backward (reset) skill, and improving learning speed and generalization. We demonstrate that this approach leads to sample-efficient learning of generalizable policies with a significant reduction in the need for manual interventions (i.e., resets).

Of course, humans and animals that must also deal with the real world also need to handle each of these challenges. However, in contrast to standard robotic RL settings, humans do not approach each new problem with a blank slate: we leverage prior experience to help us acquire new skills quickly, scaffold the process of learning that skill (e.g., by using skills we already have to practice and try again), and utilize representations learned from prior experience to ensure that the new skill is represented in a robust and generalizable way. We study an analogous approach in this work: while prior work has discussed many of the challenges associated with real world robotic learning in isolation, and proposed individual technical solutions [10, 19, 17, 61, 56, 16, 48, 35, 25, 26, 3, 38], we instead ask whether all of these issues can be alleviated by appropriately incorporating prior data. Our central thesis is that prior experience that the robot collected for performing a variety of other related tasks can facilitate many seemingly distinct aspects of autonomous robotic learning, including better sample-efficiency, better generalization of skills learned in a narrow setting, and learning with infrequent resets.

The strategy of accelerating learning by initially pretraining on a broad prior dataset is supported by the widespread success of pretraining and finetuning in supervised learning. Prior work in computer vision and natural language processing has demonstrated that pretraining on large and diverse datasets enables both data efficiency and broad generalization [2, 23, 44, 1]. Thus, extending the idea of pretraining on diverse data to robotics is a natural step towards more effective real-world learning. With the goal of taking a data-driven approach to robotics, prior work has explored offline RL to leverage static datasets [42, 25, 55, 13, 30, 41, 53, 31, 38, 12, 28, 29], multi-task and meta RL to handle diverse multi-task data [54, 39, 51, 9, 21, 59, 57, 58, 50, 26, 60], and reset-free learning to collect data autonomously [10, 19, 17, 61, 56, 16, 48, 35]. However, addressing the full real-world learning problem, as we discussed before, requires not only algorithms for learning from prior data, but the right system that combines these algorithms into a multi-task, reset-free framework. To this end, we propose a complete system for extracting useful skills from prior data and applying them to learn new tasks autonomously. Although the individual components of our system are based on previously proposed principles and methods for offline RL [38], multi-task learning with task embeddings [26, 20], and reset-free learning with forward-backward controllers [19, 10, 61, 56, 48], our system combines these components into a novel framework that effectively enables real-world robotic learning and leverages prior data for all of these parts.

Our system consists of two phases: offline learning using the prior data, and mostly autonomous online finetuning on a new task (see Figure I), where only occasional resets are provided. In the offline phase, our goal is to extract skills and representations from the prior data that may be useful on the new task. Accordingly, we use an offline RL algorithm to obtain a multi-task policy capable of performing the different behaviors in the offline data. Then, in the online phase, we adapt the learned skills to the new task. To enable autonomous learning, we finetune two distinct skills for each new task, alternating between attempting the task and resetting the environment. If the behaviors required to perform the target task and reset the environment are structurally similar to those in the prior data, the skills learned offline will succeed at performing and resetting the task with non-negligible probability, providing a learning signal that allows the agent to adapt its behavior with only a small amount of online data. Moreover, initially training on a broad dataset of experience encourages learning representations that are robust to axes of variation that are irrelevant to performing the task. Where polices learned from narrow data may overfit to unimportant details of the environment and fail when those details change, polices learned from diverse multi-task data can generalize to new conditions with little to no adaptation.

Our main contribution is demonstrating that incorporating prior data into a reinforcement learning setup simultaneously addresses several key challenges in real-world robotic RL: sample-efficiency, zero-shot generalization, and autonomous non-episodic learning. Prior work has addressed parts of the real-world learning problem with specific technical solutions, but we build a complete system for robotic reinforcement learning that leverages prior data. The individual components of our system, such as the choice of RL algorithm, are not novel, but the combination of these parts is a system with novel capabilities to bootstrap effective real-world online RL with multi-task prior data. We validate our approach on real-world robotic manipulation, where we show that our method makes it possible to learn to manipulate new objects via RL in settings where prior approaches struggle to make progress, reach final performance that is comparable to what the algorithm can attain when provided with "oracle" demonstration data for the new task, and acquire policies that generalize more broadly than those trained only on single-task data.

II. Related Work

In this section we review prior work that has explored learning behaviors from offline datasets, leveraging multi-task data, and reinforcement learning with minimal resets. We build on techniques from each area to devise a complete system for automated robot learning.

Accelerating online RL with offline data. Our method is not the first to propose learning behaviors from offline data to improve the efficiency of RL. Methods for imitation learning use offline data in the form of demonstrations to learn behavior, and imitation learning can be used to accelerate reinforcement learning [47, 22, 45, 40, 52, 27, 15, 37]. However, when the offline data is sub-optimal, offline reinforcement learning methods typically perform better [42, 25, 55, 13, 30, 41, 53, 31, 38, 12, 28, 29] and can also be combined with online finetuning [38, 24, 26]. We use an offline RL method to extract useful skills and representations from prior data, but our system includes additional components that allow us to use multi-task offline data and autonomously finetune the policy online by learning to both perform the task and reset.

Multi-task and meta-RL. Offline RL methods typically assume the data is labeled with rewards for a single task. Methods for multi-task RL [54, 39, 51, 9, 21, 59, 57, 58, 50, 26, 60] address the additional challenges that arise when learning from multi-task data, like sharing model parameters between tasks. Some approaches share parameters by learning a single policy conditional on a space of tasks [26, 20]. We similarly adopt the strategy of learning a task conditioned policy, however unlike multi-task RL methods our focus is adapting this multi-task policy to unseen tasks. Meta-RL methods [8, 11, 46, 62] share our goal of using data from prior tasks to quickly adapt to new tasks, but typically operate in the online setting. We only assume access to offline data from previous tasks. Offline meta-RL methods [32, 7, 43, 36] are capable of using offline data, but do not address adaptation with minimal resets like our system. Perhaps the closest prior system to ours in this area is Kalashnikov et al. [26], which studies multi-task learning for a similar class of grasping-based robotic manipulation tasks. However, in contrast to this prior work, our focus is not on how to train a multi-task policy, but specifically on how prior offline data from varied tasks can be leveraged to improve the autonomy and generalization when learning a new task. While Kalashnikov et al. [26] also evaluates finetuning to new tasks, our system goes significantly further, aiming to automate resets (which the prior work does not address, instead using an instrumented bin setup), and evaluating the benefits in terms of generalization and efficiency for the new task.

Reset-free RL. Our system aims to utilize prior data to autonomously learn robotic skills, leveraging prior experience to both accelerate the learning of a new behavior and the process of learning how to *reset* between attempts. Prior work has tackled this "reset-free" learning problem [49] in a number of ways. Some techniques exploit the the observation that when learning several tasks simultaneously, some tasks reset others, thus forming a curriculum [16, 35, 17, 19]. While Gupta et al. [16] also uses a multi-task setup to automate resets, this prior paper involves a small number of tasks that all focus on enabling reset-free learning for a particular (single) skill, whereas our focus is specifically on using prior data for *other* tasks to enable a *new* task to be learned as autonomously as possible. Other work learns separate controllers for performing the task and resetting the environment [19, 10, 61, 56, 48]. We also learn separate controllers, but initialize them with potentially useful skills extracted from prior data. Most similar to our work, Sharma et al. [48] use prior data in the form of demonstrations to accelerate learning forward and reset behaviors. However, our system does not rely on expert demonstrations of the target task. Instead, we use potentially sub-optimal data from other tasks.

III. Preliminaries

The goal in standard reinforcement learning (RL) is to learn a policy $\pi(\mathbf{a}|\mathbf{s})$ that maximizes the long-term cumulative discounted reward in a Markov decision process (MDP), which is defined as a tuple $\mathcal{M} := (\mathcal{S}, \mathcal{A}, P, r, \rho, \gamma)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$ is the transition function, $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, $\rho \in \Delta(\mathcal{S})$ is the initial state distribution, and $\gamma \in [0, 1]$ is the discount factor. The objective in standard RL is to optimize the policy against the cumulative discounted return objective, starting from the initial state distribution ρ:

$$J_{\text{standard}}(\pi) := \mathbb{E}_{\substack{\mathbf{s}_0 \sim \rho, a_i \sim \pi, \\ \mathbf{s}'_{i+1} \sim P(\cdot|\mathbf{s}_i, \mathbf{a}_i)}} \left[\sum_i \gamma^i r(\mathbf{s}_i, \mathbf{a}_i) \right] \quad \text{(Standard)}. \tag{1}$$

The Q-function $Q^\pi(\mathbf{s}, \mathbf{a})$ for a policy $\pi(\mathbf{a}|\mathbf{s})$ is the long-term discounted reward obtained by executing action $\mathbf{a}$ at state $\mathbf{s}$ and following $\pi(\mathbf{a}|\mathbf{s})$ thereafter. $Q^\pi(\mathbf{s}, \mathbf{a})$ is the fixed point of $Q(\mathbf{s}, \mathbf{a}) := R(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim P(\cdot|\mathbf{s}, \mathbf{a}), \mathbf{a}' \sim \pi(\cdot|\mathbf{s}')} [Q(\mathbf{s}', \mathbf{a}')]$, which is known as the Bellman equation. While the standard RL objective can be optimized in several ways, in this work we build on the class of actor-critic methods [33, 18, 29]. A typical actor-critic method alternated between two phases, policy evaluation and policy improvement. In the policy evaluation phase the current policy $\pi(\mathbf{a}|\mathbf{s})$ is evaluated by fitting an action-value function $Q^\pi(\mathbf{s}, \mathbf{a})$. In the policy improvement phase, the policy or actor $\pi(\mathbf{a}|\mathbf{s})$ is updated to maximize the expected Q-value.

Autonomous RL. The standard RL objective is typically optimized in an "episodic" manner, where between each trial, the agent is reset into an initial state sampled indendently from $\rho(s)$. However, in this paper, we study the problem of reinforcement learning with minimal external resets, also known as autonomous reinforcement learning [49]. This requires the policy to be trained without resetting after each trial (though we do use occasional resets in our experiments). This is important in real-world settings, where it may take considerable effort to provide manual resets after every single attempt. Sharma et al. [49] formalize autonomous RL in terms of two MDPs: a non-episodic training MDP, $\mathcal{M}_T$, where the environment resets only occasionally or never at all, and the corresponding episodic evaluation MDP $\mathcal{M}$, where the agent must successfully perform the task from initial states sampled from $\rho(s)$. The transition dynamics P and the reward function r are the same in the two MDPs. The agent must learn to perform the task in $\mathcal{M}_T$, but will be evaluated in terms of its performance on $\mathcal{M}$. Since only infrequent resets are allowed during training in $\mathcal{M}_T$, simply maximizing the original reward $r(\mathbf{s}, \mathbf{a})$ in $\mathcal{M}_T$ may not lead to the best performance on $\mathcal{M}$ [49], and so the agent must learn to reset itself while training so that it can learn to attain good performance from the initial states it will encounter at test-time in $\mathcal{M}$.

Offline reinforcement learning via AWAC. In this paper, we utilize an offline RL method, advantage-weighted actor critic (AWAC) [38], to extract skills from prior data. AWAC is an actor-critic method that alternates between a policy evaluation phase (Equation 2), where it trains a parameteric Q-function, $Q_\phi(\mathbf{s}, \mathbf{a})$, to minimize the error between two sides of the Bellman equation on the samples from the offline dataset, and a policy improvement phase (Equation 3) where it improves a parametric policy $\pi_\theta(\mathbf{a}|\mathbf{s})$ via advantage-weighted updates. Advantage-weighted updates perform policy improvement by cloning actions that are highly advantageous under the learned Q-function and are hence, more likely to improve upon the data-collection policy. Denoting the offline dataset as $\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}_{i=1}^N$, where $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$ denotes a single transition, the training objectives for AWAC are given by:

$$\widehat{Q}^\pi_\phi \leftarrow \arg\min_\phi \; \mathbb{E}_{\mathbf{s}, \mathbf{a}, \mathbf{s}' \sim \mathcal{D}} \left[(Q_\phi(\mathbf{s}, \mathbf{a}) - (r + \gamma Q_\phi(\mathbf{s}', \mathbf{a}')))^2 \right] \tag{2}$$

$$\widehat{\pi} \leftarrow \arg\max_\theta \; \mathbb{E}_{\mathbf{s}, \mathbf{a} \sim \mathcal{D}} \left[\log \pi_\theta(\mathbf{a}|\mathbf{s}) \cdot \exp(\widehat{A}^\pi(\mathbf{s}, \mathbf{a})) \right],$$
$$\text{where} \quad \widehat{A}^\pi(\mathbf{s}, \mathbf{a}) := \widehat{Q}^\pi_\phi(\mathbf{s}, \mathbf{a}) - \mathbb{E}_{\mathbf{a}' \sim \pi} \left[\widehat{Q}^\pi_\phi(\mathbf{s}, \mathbf{a}') \right]. \tag{3}$$

IV. Problem Statement

In this section, we will formalize our problem statement and describe our evaluation setup. Our goal is to utilize data from previous tasks to address three key challenges in robotic reinforcement learning: sample-efficiency, autonomous learning and generalization and robustness to unseen test conditions, all together. To this end, we assume access to a prior dataset $\mathcal{D}_{\text{prior}}$ of experience collected from a multiple training tasks, $\Pi = \{\tau_1, \tau_2, \cdots, \tau_N\}$:

$$\mathcal{D}_{\text{prior}} = \cup_{j=1}^N \mathcal{D}_j; \quad \mathcal{D}_j = \left\{ \left(\mathbf{s}_i^j, \mathbf{a}_i^j, \mathbf{s}_i'^j, r_i^j \right) \right\}_{i=1}^K, \tag{4}$$

where $\mathcal{D}_j$ denotes the chunk of the prior data corresponding to task τ_j. Our goal will be to extract a rich repertoire of skills from this diverse, multi-task prior dataset, $\mathcal{D}_{\text{prior}}$, and use them to quick solve a new target task τ^* that was not seen in training, i.e., $\tau^* \notin \Pi$. Additionally, we require that this new task must be solved with minimal resets: when the robot is practicing on the target task τ^*, it will only be infrequently provided with external resets, in accordance with the autonomous RL paradigm. The learned agent is then evaluated on the same task τ^*, but this time it gets reset after every episode.

V. Autonomous Robotic Reinforcement Learning with Prior Data

We will now present our approach for leveraging data from previous tasks to address key challenges in robotic reinforcement learning: sample-efficiency, non-episodic learning, and generalization. Our system is designed to extract knowledge from $\mathcal{D}_{\text{prior}}$ that enables learning new tasks more efficiently, with fewer resets, and in a way that results in broader generalization to variations in the environment.

Our method, which we call Autonomous RobotIc REinforcement Learning (ARIEL), consists of two phases (see Figure I). First, we extract useful skills and representations from $\mathcal{D}_{\text{prior}}$ using offline reinforcement learning, as discussed in Section V-A. This essentially provides our method with the "prior knowledge" that it will use to then perform near-autonomous training for new tasks. We use this prior knowledge to both quickly learn the new task, and to bootstrap a separate policy that resets the environment back to an initial state, so that the agent can continue practicing without the need for external resets after every attempt.

A. Learning From Prior Data with Offline RL

Given a dataset from previous tasks, we first aim to extract skills and representations that could be useful for learning downstream tasks. The multi-task dataset $\mathcal{D}_{\text{prior}}$ can come from many different sources: human demonstrations, sub-optimal data from previous reinforcement learning experiments, or even data collected using (imperfect) hand-engineered policies. We assume that the tasks in this dataset are structurally similar to the new tasks we wish to learn, so a (multi-task) RL agent that has been trained on this dataset should explore new environments much more effectively than a randomly initialized policy. For example, the prior data might of consist picking and placing a variety of objects under a wide variety of lighting conditions and backgrounds. An agent that successfully learns the skill of picking and placing from this prior data would then be well-prepared to learn how to pick up a previously unseen object, and place it in a previously unseen container.

The provided prior dataset $\mathcal{D}_{\text{prior}}$ is partitioned into individual task datasets (Equation 4), and to make use of this multi-task dataset for downstream tasks, we train a conditional policy $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{z})$ using offline RL. This policy is conditioned on a task index $\mathbf{z}$ (represented as a one-hot vector) that informs the agent which task it is performing. Conditioning on task indices is a common strategy while training multi-task policies [26]. Training a single multi-task policy offers several advantages over training separate policies on the prior dataset. First, we are able to learn a representation of policy inputs and outputs (in this case, images and desired change in end-effector pose, respectively) which is shared across tasks, and is therefore more likely to generalize well when faced with new inputs: both through interpolation (if the new task is similar to previously seen tasks), and extrapolation (for novel tasks). Second, when attempting a new task, we only need to sample actions from a single pre-trained policy, as opposed to choosing from an ensemble of pre-trained policies. Finally, this also allows our method to gracefully scale with the number of tasks in the prior dataset. After the offline learning phase, we have a multi-task policy $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{z})$ that can perform each of the tasks in the prior data. In Section VI we discuss the specific tasks we use in our prior data and how we generate the prior dataset for our experiments.

B. Autonomous Online Training

The main goal of our system is to enable the robot to efficiently learn a new task with minimal externally provided environment resets, leveraging the prior knowledge distilled into the offline multi-task policy discussed in the previous section. Since the environment is not frequently reset, the agent must simultaneously learn to perform the task and reset the environment to the initial state so it can continue practicing. Prior work has tackled this problem by learning distinct forward and backward controllers, alternating between applying the forward controller to perform the task and applying the backward controller to reset the task [19, 10, 61, 56, 48]. We adopt an analogous strategy, but we leverage the multi-task policy described in the previous section to bootstrap learning in both the forward and backward direction, which both helps to overcome the exploration challenge and provide automated resets more quickly than if the backward controller were trained from scratch.

To enable this, we fine-tune the parameters of the policy $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{z})$ obtained after the offline phase with online data from the new task τ^*, that we denote as $\mathcal{D}_{\tau^*}$, similar to prior work that uses online fine-tuning to improve upon policies learned via offline RL alone [38]. However, since we have a multi-task policy, we must confront the question of which skill(s), parameterized by task indices $\mathbf{z}$, should be chosen to start finetuning for the forward and backward controller. In order to automatically determine which prior skill (or combination of prior skills) to finetune, we treat the task index $\mathbf{z}$ as a continuous, learnable parameter and use data collected during online reinforcement learning to select the $\mathbf{z}$'s best suited for the given new task. In this sense, we treat $\mathbf{z}$ as a task embedding [20] that is automatically adapted for the task at hand, similar to prior work in meta-reinforcement learning [46].

Since we aim to learn both a forward and backward controller, we optimize for two skills, parameterized by two distinct task embeddings $\mathbf{z}_f$ and $\mathbf{z}_b$, that respectively perform the task and reset the environment. Conditioning the policy on $\mathbf{z}_f$ results in the forward controller $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{z}_f)$ that performs the task, whereas conditioning the policy on $\mathbf{z}_b$ results in the backward controller $\pi_\theta(\mathbf{a}|\mathbf{s},\mathbf{z}_b)$ that resets the environment according to the initial state distribution of the task τ^*. During fine-tuning we alternate between applying the forward controller and backward controller, optimizing the forward controller with rewards for completing the task $r_f(\mathbf{s},\mathbf{a})$ and the backward controller with rewards for returning to the initial state distribution $r_b(\mathbf{s},\mathbf{a})$. We optimize the parameters $\mathbf{z}_f$ and $\mathbf{z}_r$ in tandem with the policy and value function parameters θ and ϕ. The parameters of the policy and value function are updated using using an actor-critic algorithm; we utilize the AWAC [38] algorithm in our experiments. The embeddings $\mathbf{z}_f$ and $\mathbf{z}_r$ are optimized differently, as we discuss next.

Optimizing task embeddings $\mathbf{z}_f$ and $\mathbf{z}_b$. Since our goal is to learn on the target task τ^* as quickly as possible, and with minimal resets, we need to effectively explore the environment during online training. An ideal approach that explores effectively must not prematurely commit to a particular value of $\mathbf{z}_f$ and $\mathbf{z}_b$ until the online experience clearly indicates so, and at the same time, it should be fast at collapsing to the correct values of $\mathbf{z}_f$ and $\mathbf{z}_b$, when it is very much apparent from the online experience gathered till then. While one straightforward approach to optimize $\mathbf{z}_f$ and $\mathbf{z}_b$ is to simply treat them as additional parameters and update them via gradient descent, however this requires initializing $\mathbf{z}_f$ and $\mathbf{z}_b$ to an initial guess, which means that only these randomly chosen skills will be fine-tuned. Furthermore, the *local* nature of gradient updates may simply inhibit fast adaptation of these embeddings.

Instead, we found that the cross-entropy method (CEM) [**?**] provided us with a favorable balance between exploration and exploitation. CEM provides a way to initialize the task embedding to a *distribution*, which ensures that we do not have to commit to fine-tuning any one particular skill. How is this CEM procedure performed? Since we want to optimize both the forward and backward embeddings $\mathbf{z}_f$ and $\mathbf{z}_b$, we fit two sampling distributions with CEM, $q_f(\mathbf{z})$ and $q_b(\mathbf{z})$. The distribution of tasks in the prior data $\mathcal{D}_{\text{prior}}$ provides us with a natural (and hopefully informative) prior to initialize $q_f(\mathbf{z})$ and $q_b(\mathbf{z})$. On each iteration of training, we sample a $\mathbf{z}$ from either $q_f(\mathbf{z})$ or $q_b(\mathbf{z})$ and roll out the policy conditioned on this $\mathbf{z}$ value for M steps. These M-step trajectories would essentially correspond to an episode in episodic RL, however the environment is (typically) not reset afterward. We alternate between sampling the $\mathbf{z}$ from $q_f(\mathbf{z})$ or $q_b(\mathbf{z})$ to alternate between applying the forward and reset controllers. The reward function correspondingly alternates between $r_f(\mathbf{s},\mathbf{a})$ and $r_b(\mathbf{s},\mathbf{a})$. We periodically refit $q_f(\mathbf{z})$ and $q_b(\mathbf{z})$ to the J most recently sampled $\mathbf{z}$'s that resulted in trajectories that successfully completed or reset the task, respectively. We update the policy and value function parameters θ and ϕ using the update steps from an RL algorithm as summarized in Algorithm 2. Every N steps, we provide an external reset to the environment. However, $N >> M$, so the external resets are infrequent. For example, in one of our real world tasks, $M = 20$, and $N = 400$. In theory, we could run our method without any manual resets at all, but we found a small number of manual resets necessary to recover from certain states, similarly to prior work in the area [10].

C. Algorithm Summary

To summarize, we first perform offline learning on the provided dataset $\mathcal{D}_{\text{prior}}$ (see Algorithm 1). We use offline RL to obtain a multi-task policy $\pi_\theta(\mathbf{a}|\mathbf{s},\mathbf{z})$ and value function $Q_\phi(\mathbf{s},\mathbf{a},\mathbf{z})$. Then, in the online phase (see Algorithm 2), we adapt this policy and value function to a new task. Notably, the offline phase occurs only once, but the online phase can be run many times to efficiently learn many new tasks. Online adaptation begins by fitting two CEM sampling distributions, $q_f(\mathbf{z})$ and $q_b(\mathbf{z})$, which are initialized to the distribution of task indices seen during offline learning. Then, we collect data for the new task by sampling a task embedding $\mathbf{z}$ from a CEM model and rolling out the policy conditioned on this $\mathbf{z}$ for M steps. If the trajectory obtains reward for completing or resetting the task, we switch to sampling from the opposite CEM model. This corresponds to alternating between applying a forward and backward controller. We use the online data to refine the sampling distributions $q_f(\mathbf{z})$ and $q_b(\mathbf{z})$ with CEM and the policy and value function with RL. Every N steps we manually reset the environment, sampling a state from the initial state distribution $s \sim \rho(s_0)$. Our system allows us to learn the new task efficiently and with minimal environment resets by leveraging prior data from previous tasks.

Algorithm 1 ARIEL offline phase

```
Input: offline dataset $\mathcal{D}_{\text{prior}}$
  Initialize policy $\pi_\theta(\mathbf{a}|\mathbf{s},\mathbf{z})$, value function $Q^\pi_\phi(\mathbf{s},\mathbf{a},\mathbf{z})$
  $\pi_\theta(\mathbf{a}|\mathbf{s},\mathbf{z}), Q^\pi_\phi(\mathbf{s},\mathbf{a},\mathbf{z}) \leftarrow \texttt{OfflineRL}(\mathcal{D}_{\text{prior}})$
```

D. Implementation Details

We choose advantage-weighted actor-critic (AWAC) as the underlying RL algorithm for both the offline and online phase because it addresses the specific challenges that arise when learning behaviors offline and then finetuning them online [38]. We present a technical description of AWAC in Section III and provide the specific hyperparameters that we use in our experiments in the Appendix. Notably, since AWAC is an off-policy RL algorithm, we have the option of continuing to train on the prior data during the online phase. We hypothesize that including the prior data during online adaptation prevents the agent from overfitting to the new task and improves generalization, so we continue to train on the prior data even during the online fine-tuning phase.

As our goal is to make robotic learning as autonomous as possible, we use images observations as part of our input to the RL agent. Image observations allow our system to learn from a diverse range of different tasks without needing to hand-engineer state representations suitable for each task. Because we are learning from images we use convolutional neural networks to model the policy and Q-function (see Figure 2). We provide the specific architecture in the Appendix. The observations additionally consist of the state of the robot's

Algorithm 2 ARIEL online adaptation

```
Input: pre-trained policy $\pi_\theta(\mathbf{a}|\mathbf{s},\mathbf{z})$ and value function $Q^\pi_\phi(\mathbf{s},\mathbf{a},\mathbf{z})$
 1: Initialize buffer $\mathcal{B}$, CEM models $q_f, q_r$
 2: Fit $q_f(\mathbf{z})$ and $q_b(\mathbf{z})$ to offline task indices
 3: $d \leftarrow f$   // task direction, f is forward, b is backward
 4: while not done do
 5:     $s \sim \rho(\mathbf{s}_0)$   // sample initial state (i.e external reset)
 6:     for $N$ steps do
 7:         // sample new task embedding if you reach max episode length
 8:         if steps % $M == 0$ then
 9:             $\mathbf{z} \sim q_d$
10:         end if
11:         Sample $\mathbf{a} \sim \pi(\cdot|\mathbf{s},\mathbf{z})$, observe $\mathbf{s}' \sim p(\cdot|\mathbf{s},\mathbf{a})$
12:         $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\mathbf{s},\mathbf{a},r_d(\mathbf{s}',\mathbf{a}),\mathbf{s}',\mathbf{z}\}$
13:         if $r_d(\mathbf{s}',\mathbf{a})$ then
14:             switch($d$)   // switch task direction
15:         end if
16:         update $\pi_\theta, Q^\pi_\phi$ with RL
17:         update $q_f(\mathbf{z}), q_b(\mathbf{z})$ with CEM
18:         $\mathbf{s} \leftarrow \mathbf{s}'$
19:     end for
20: end while
```

joints, and the task embedding $\mathbf{z}$. We use a Gaussian mixture model as the sampling distribution for CEM. For our evaluations in simulation we fit the CEM sampling distributions to the $J = 25$ most recent successful embeddings. For our evaluations in the real world, we fit the CEM models to the $J = 10$ most recent successful embeddings.

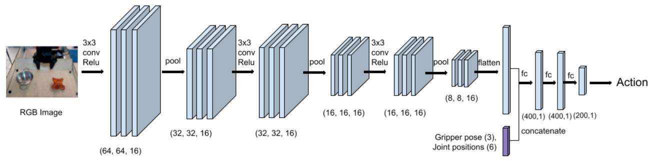

Fig. 2. **CNN architecture we use for our policy.** Our method learns robotic manipulation skills from raw image inputs, and fine-tunes to new skills with minimal resets while leveraging prior experience.

VI. Experiment Setup

We now describe the system and experiment setup in our evaluation, as well as the data collection and training process.

Robotic platform. We evaluate our approach using a 6-DoF WidowX robotic arm. The 7D action space of the system consists of 6D Cartesian end-effector motion, corresponding to relative changes in pose, as well as one dimension to control the opening and closing of the parallel jaw gripper. The state observations consist of the joint angles and RGB images from an overhead camera, which are 48×48 in simulation and 64×64 in the real world.

Evaluation tasks. The tasks, some of which are shown in Fig. 4, involve interacting with objects and either containers or drawers, with each task corresponding to a different object

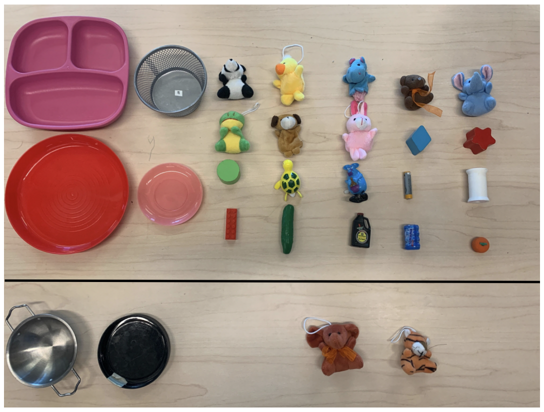

Fig. 3. **Objects used to construct the container placing tasks in our experiments.** The upper part of the figure shows containers and objects used in the offline data for pretraining, the lower part shows test objects that are used as part of new tasks for online training.

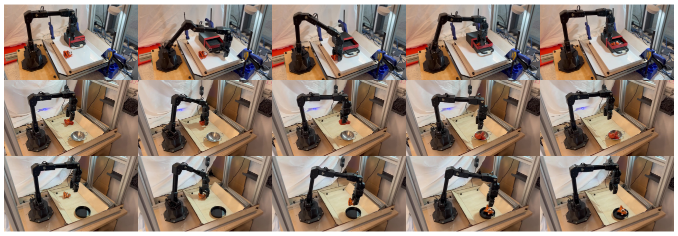

Fig. 4. Three downstream tasks on our real-world robotic setup. Each row shows snapshots from a single episode, from left to right. The top row depicts the drawer task, where the objective is to open the drawer, and place an object inside the drawer. The middle and bottom rows depict different container tasks, where the goal is pick an object from the workspace, and place it either on a plate, or inside a pot, depending on the task.

and a different container. For container picking and placing tasks, there are 20 possible training objects and 4 possible training containers, including plates and cups (see Fig. 3, and 2 unseen test objects and 2 unseen test containers. The prior data for the offline phase consists of 20 forward and 20 backward tasks with random object-container combinations. The simulated experiments have 16 possible objects and 1 container. For the drawer tasks, there are 4 possible training objects and 1 unseen test object, corresponding to 4 forward and 4 backward tasks. When a container is present, the tasks involve picking up an object from the table and placing it inside the container, whereas the drawer task additionally requires closing the drawer after inserting the object, and resetting requires opening the drawer and taking the object out. The new test tasks that we finetune in the online phase involve test objects that were not seen in training, but the same types of picking and placing behaviors. If indicated, the new task may also include a new type of container, as shown in Fig. 3. All tasks have sparse rewards evaluated using a simple hand-designed vision system, with a reward of +1 if the robot completes the forward (or backward) task, and 0 otherwise. While such rewards are easy to define, they present a substantial challenge for RL.

Generating prior data. Our system can use prior data from any reasonable source, including demonstrations and prior RL runs, but for simplicity and consistency we employ a simple scripted policy to collect the prior data that we use for the offline phase. This provides a largely autonomous strategy to collect mediocre data for a variety of objects. While this scripted policy fails very frequently, offline RL can make use of such mediocre data effectively to pretrain value functions and policies for downstream fine-tuning. Since the prior dataset needs to provide an initialize for both the forward and backward directions for new tasks, we perform scripted collection in a reset-free manner, with the robot attempting to move the objects both into and out of containers. We collected about 500 trajectories for each of the 40 (20 forward, 20 backward) container pick and place tasks, and about 150 trajectories for each of the 8 (4+4) object/drawer tasks. All trajectories have a length of 30 time steps. The average success rates of the scripted policy were 0.38 for the simulated object-container tasks, 0.35 for the real-world object-container tasks, and 0.93 for the real-world object-drawer tasks.

Online training setup. Online training includes infrequent resets, to address the case where an object becomes stuck in a hard-to-reach (or unreachable) part of the workspace, but otherwise requires the robot to train autonomously. The resets are provided every 80 trials in simulation (about every hour), about every 20-30 trials (or about every 20-30 minutes) in the real world. Additionally, the pose of the robot is moved to a neutral position at the beginning of every trial, since there is no physical limitation that prevents doing this automatically, but the robot must still handle the fact that the objects in the scene maintain their position across trials.

VII. Results

Our experiments aim to evaluate the hypothesis that leveraging prior data from other tasks can simultaneously address multiple challenges in autonomous real-world robotic RL. To this end, we evaluate ARIEL on a suite of simulated robotic manipulation tasks, and evaluate real-world online training with three new downstream tasks involving objects not seen in the prior offline data. Our goal is to answer the following questions: **(1)** Does leveraging prior data enable mostly autonomous learning of a new task in simulation and in the real-world? **(2)** How does ARIEL compare to prior methods for leaning with minimal resets that do not use prior data? **(3)** Does leveraging prior data via ARIEL lead to gains in sample-efficiency for a new task? and **(4)** Does ARIEL produce policies that better generalize to new conditions?

Videos of the experiments can be found on the anonymous website: https://sites.google.com/view/ariel-paper/

A. Evaluating ARIEL in Simulation

We begin our empirical evaluation by comparing ARIEL to prior approaches that also aim to tackle the challenges of autonomous learning or sample-efficient learning, in simulation.

*1) **ARIEL enables effective autonomous learning with minimal resets**:* We compare to two prior approaches in our simulated experiments. To comparatively evaluate how well our method enables learning with minimal resets, we compare to the perturbation controller approach in **R3L** [61], which alternates between training a forward controller to optimize a task-completion reward and training a perturbation controller that optimizes a novelty exploration bonus. This method has been proposed specifically to handle reset-free learning for real-world RL problems. We initialize the forward controller in R3L with the policy obtained by running offline multi-task RL on the prior data that used by our approach. We also compare to a method that does *not* optimize the task embeddings $\mathbf{z}_f$ and $\mathbf{z}_b$ over the course of autonomous online fine-tuning. We refer to this approach as **Multi-task RL**. This is equivalent to first running multi-task offline RL on the prior dataset, and then fine-tuning the policy using a fixed task index (that was unused during pre-training) in an online phase afterwards. This method is conceptually similar to MT-Opt [26], but adapted to our pre-training and then fine-tuning setup, as opposed to the re-training from scratch approach followed by Kalashnikov et al. [26]. For instructive purposes, we also compare to an "oracle" version of our approach, labeled (**ARIEL + resets**), which assumes access to external resets at the end of each episode. While resetting every episode is prohibitively expensive in the real-world, we can still run this method in simulation for our understanding.

The results in Figure 5 show that ARIEL is the only approach (and its oracle reset variant) are the only methods that succeed at learning the new tasks, and the full ARIEL method closely matches final oracle performance. While the poor performance of prior methods might be surprising, recall that these tasks require using raw image observations and sparse 0/1 rewards, which present a significant challenge for any RL approach. While **R3L** and **Multi-Task RL** both succeeded at learning the tasks in the offline phase (see the Appendix for the offline RL learning curves), they are unable to make progress during online training of the new task. The comparison to **Multi-task RL** indicates the importance of adapting the task embeddings: if the task embeddings are not adapted to the new task, distributional shift in the task embeddings may severely hamper effective reset-free learning. Thus, we find that these methods cannot use skills learned offline to efficiently explore the new task. Our method solves this challenge with an explicit mechanism for searching in the space of task embeddings, starting from those used during offline learning, to induce substantially better-than-random initial exploratory behavior.

*2) **ARIEL enables sample-efficient learning of the new task**:* Next, we aim to evaluate whether our method, ARIEL, enables better sample-efficiency than learning from scratch. To study the effect on sample complexity in isolation, we conduct experiments in which the environment is reset at the end of every episode (as opposed to the infrequent manual reset setting studied in the rest of this paper). We compare our method to simply applying RL to the target task, using the same RL algorithm as our system (**AWAC (no prior data)**). We also compare to an **oracle** method which is trained offline on data coming from only the single new task of interest and then trained online on the same task. Perhaps unsurprisingly, we found that learning from scratch fails in the challenging

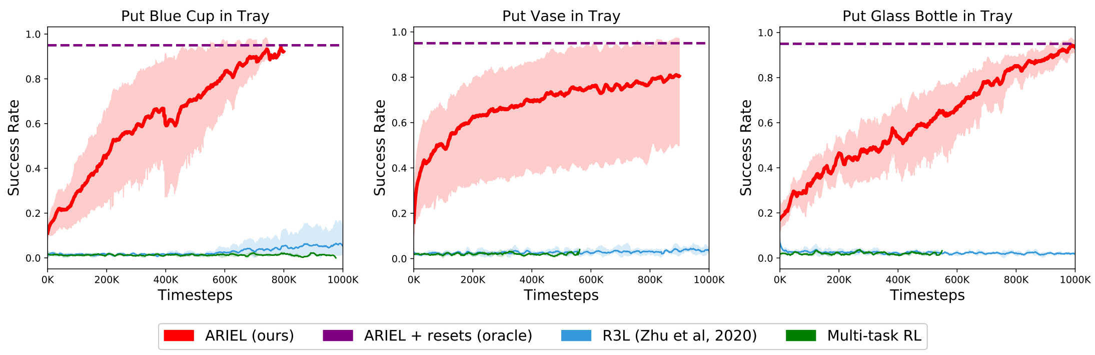

Fig. 5. Experiments on simulated domains comparing our method to prior works in reset-free and multi-task learning. We see that ours is the only method that is able to learn in this limited-reset setting.

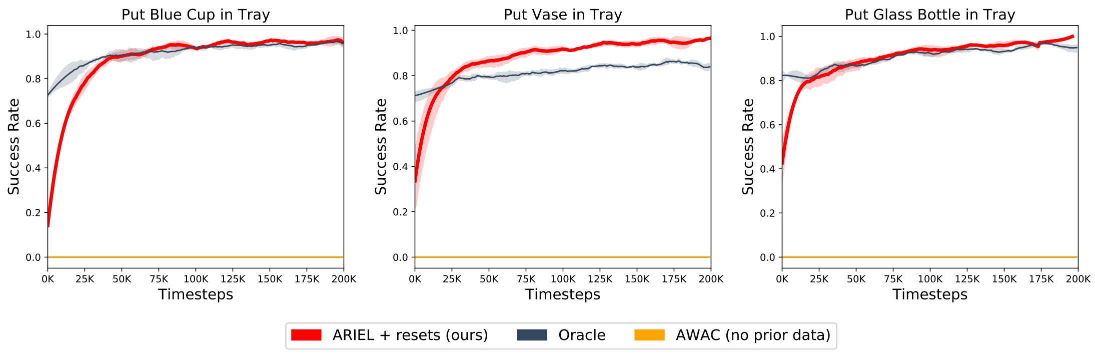

Fig. 6. Experiments in simulated domains comparing our method to learning without any prior data, and against an oracle approach which assumes access to offline dataset from the downstream task of interest. We see that our method adapts almost as quickly as the oracle approach, whereas reinforcement learning from scratch fails to make any progress. Note that this experiment is not in a limited-reset settings, resets were conducted at the end of every episode.

setting of learning from images and sparse rewards. However, our method learns just as efficiently as the oracle despite only having access to offline data for other tasks and not the new task of interest. Thus, our method demonstrates that properly leveraging prior data can improve sample-efficiency even when data is only available for other tasks.

B. *Real-World Experiments*

Next we evaluate the performance of ARIEL when training online to solve new instances of the pick and place and drawer tasks, with previously unseen objects, in the real world. We aim to study how offline data can enable near-autonomous real-world RL with sparse rewards, and also study question **(4)**, understanding the effect that the prior data has on the generalization of the resulting policies. After pretraining on the datasets discussed in Section VI, we evaluate ARIEL by training three tasks that involve objects not seen in the prior data: placing a toy tiger in a drawer (which requires also closing the drawer, then opening it again to reset), placing a toy elephant in a pot, and placing a toy tiger on a lid. The pot, lid, elephant, and tiger were not seen in the prior data. We evaluate success rates of the policy learned in the offline phase, the policy learned after 100 trials of online training, and the final policy after fine-tuning for 600 trials.

*1) **Autonomous fine-tuning with infrequent resets with ARIEL significantly improves performance in the real world**:* Observe in Tables I and II that the performance of both the forward and backward controllers increases moderately after 100 trials (e.g., from 1/10 to 2/10 for the forward controller and from 1/10 to 8/10 for the backward controller on put elephant in pot) and substantially by the end of online training (e.g., from 2/10 to **7/10** for the forward controller and 8/10 to **10/10** for the backward controller) indicating that ARIEL successfully optimizes the embeddings and policy parameters, with minimal resets, to perform the new task. Since prior autonomous training methods did not make any learning progress in our simulated experiments, we did conduct additional real-world comparisons, but instead use the real-world setup to confirm that our method can enable real-world training, and then use it to study generalization as discussed below.

*2) **Training on diverse prior data improves zero-shot generalization in the real world**:* In our final experiment, we aim to understand whether initializing from prior data provides ARIEL with more robust representations that facilitate learning policies that generalize more effectively to new conditions with any further adaptation. In principle, if our only goal is to attain good performance on a single task, the best possible source of data should come from that task itself. But in the real world, we might often instead prefer a policy that is robust to variation in the task. In this case, we might hypothesize that the larger diversity present in the prior data might provide ARIEL with representations that, even after online training on a single task, lead to better generalization.

To study this hypothesis, we additionally evaluate the fine-

	Put Tiger in Drawer	Put Elephant in Pot	Put Tiger on Lid
Offline only	2/10	1/10	0/10
100 Trials	4/10	2/10	4/10
600 Trials	**9/10**	**7/10**	**7/10**

TABLE I
REAL-WORLD EVALUATION OF THE FORWARD CONTROLLER IN THE CONTAINER AND DRAWER SETTING. FINETUNING IS MOSTLY AUTONOMOUS WITH A RESET EVERY 30 TRIALS.

	Put Tiger in Drawer	Put Elephant in Pot	Put Tiger on Lid
Offline Learning	5/10	1/10	2/10
100 Trials	6/10	8/10	**6/10**
600 Trials	**7/10**	**10/10**	**6/10**

TABLE II
REAL-WORLD EVALUATION OF THE BACKWARD CONTROLLER IN THE CONTAINER AND DRAWER SETTING. FINETUNING IS MOSTLY AUTONOMOUS WITH A RESET EVERY 20 TRIALS.

tuned policies for the *tiger in drawer* and *tiger on lid* tasks in a setting where we swap out the object (but keep the same container). We do not expect the policies to succeed consistently in this case, but we see in Tables III and IV that the policies learned by ARIEL exhibit some generalization, attaining an average success rate of **61.67%** when evaluated at Epoch 15. We include a baseline that uses only the data for each online task (i.e., either only *tiger in drawer* or only *tiger on lid*) and no prior data. This baseline only succeeds in 26.67% of trials on the new objects. This suggests that pretraining on prior data does effectively boost generalization.

Interestingly, we observe that while the success rate of ARIEL during fine-tuning improves as more steps are performed (see 100 trials vs 600 trials in Tables III and IV), this also adversely affects the zero-shot generalization performance on a new object. This is to be expected, since even though our online training procedure replays the prior data, we train with a larger proportion of online data so with sufficient training we expect the policy to become more specialized to the fine-tuning task. This observation is also consistent with prior RL works [5, 14] that note a drop in performance on test tasks as more updates are made on the training tasks. We anticipate the more principled early stopping or replay of the prior data could alleviate this issue, which is an interesting avenue for future work.

VIII. Discussion

In this work, we proposed overcoming the numerous challenges of real-world robotic reinforcement learning through better leveraging prior datasets. While reinforcement learning often involves acquiring a skill from scratch, such a formalism is not well-suited to real world conditions where running online data collection is expensive, and previously collected datasets are often accessible. While the use of previously collected datasets for tackling a particular challenge (such as that of sample efficiency [38], or better generalization [34]) has been addressed in prior work, our work aims to address such challenges jointly in a single robotic learning system, which leverages prior data as the primary mechanism to facilitate autonomy during online training, improve efficiency, and boost generalization. In our experiments, we show that our method is able to make consistent progress in online finetuning despite having to learn from sparse rewards, matching or exceeding the final performance of an oracle baseline that receives prior data *of the actual test task*, and significantly outperforming a baseline without prior data. This supports our central hypothesis that prior data can be a significant facilitator for real-world robotic RL. One limitation of our analysis is that our new online training tasks are structurally quite similar to the prior data used for offline pretraining. While the objects during online training are different, they do not require significantly distinct physical skills. We believe that, as the breadth and diversity of the available prior data increases, the offline pretrained model will correspondingly be useful for a broader range of new tasks. Exploring this direction by scaling up the proposed approach is an exciting avenue for future work.

	Our Method (ARIEL)			No Prior Data
Number of trials →	100	360	600	Best Epoch
Monkey	5/10	6/10	5/10	6/10
Rabbit	2/10	4/10	0/10	0/10
Hippo	3/10	5/10	0/10	0/10
Tiger (fine-tuning)	2/10	6/10	6/10	7/10

TABLE III
GENERALIZATION RESULTS, ZERO-SHOT PERFORMANCE ON NEW TASK OF PICKING SEVERAL OBJECTS UP AND PLACING THEM IN A CONTAINER.

	Our Method (ARIEL)			No Prior Data
Number of trials →	100	360	600	Best Epoch
Pickle	3/5	5/5	3/5	1/5
Turtle	1/5	4/5	2/5	2/5
Dog	2/5	2/5	4/5	2/5
Tiger (fine-tuning)	3/5	3/5	3/5	1/5

TABLE IV
GENERALIZATION RESULTS, ZERO-SHOT PERFORMANCE ON OPENING A DRAWER AND PLACING SEVERAL OBJECTS.

References

[1] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[2] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[3] Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, Mel Vecerik, et al. Scaling data-driven robotics with reward sketching and batch reinforcement learning. *arXiv preprint arXiv:1909.12200*, 2019.

[4] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012, 2015.

[5] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR, 2020.

[6] E Coumans and Y Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. *GitHub repository*, 2016.

[7] Ron Dorfman and Aviv Tamar. Offline meta reinforcement learning. *arXiv e-prints*, pages arXiv–2008, 2020.

[8] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[9] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pages 1407–1416. PMLR, 2018.

[10] Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. *arXiv preprint arXiv:1711.06782*, 2017.

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[12] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *arXiv preprint arXiv:2106.06860*, 2021.

[13] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019.

[14] Dibya Ghosh, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P Adams, and Sergey Levine. Why generalization in rl is difficult: Epistemic pomdps and implicit partial observability. *Advances in Neural Information Processing Systems*, 34, 2021.

[15] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

[16] Abhishek Gupta, Justin Yu, Tony Z Zhao, Vikash Kumar, Aaron Rovinsky, Kelvin Xu, Thomas Devlin, and Sergey Levine. Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention. *arXiv preprint arXiv:2104.11203*, 2021.

[17] Sehoon Ha, Peng Xu, Zhenyu Tan, Sergey Levine, and Jie Tan. Learning to walk in the real world with minimal human effort. *arXiv preprint arXiv:2002.08550*, 2020.

[18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[19] Weiqiao Han, Sergey Levine, and Pieter Abbeel. Learning compound multi-step controllers under unknown dynamics. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6435–6442. IEEE, 2015.

[20] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

[21] Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3796–3803, 2019.

[22] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, et al. Learning from demonstrations for real world reinforcement learning. 2017.

[23] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

[24] Ryan Julian, Benjamin Swanson, Gaurav S Sukhatme, Sergey Levine, Chelsea Finn, and Karol Hausman. Efficient adaptation for end-to-end vision-based robotic manipulation. 2020.

[25] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

[26] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

[27] Petar Kormushev, Sylvain Calinon, and Darwin G Caldwell. Robot motor skill coordination with em-based reinforcement learning. In *2010 IEEE/RSJ international conference on intelligent robots and systems*, pages 3232–3237. IEEE, 2010.

[28] Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pages 5774–5783. PMLR, 2021.

[29] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. 2021.

[30] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *arXiv preprint arXiv:1906.00949*, 2019.

[31] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

[32] Jiachen Li, Quan Vuong, Shuang Liu, Minghua Liu, Kamil Ciosek, Keith Ross, Henrik Iskov Christensen, and Hao Su. Multi-task batch reinforcement learning with metric learning. *arXiv preprint arXiv:1909.11373*, 2019.

[33] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[34] Yen-Chen Lin, Andy Zeng, Shuran Song, Phillip Isola, and Tsung-Yi Lin. Learning to see before learning to act: Visual pre-training for manipulation. *CoRR*, abs/2107.00646, 2021.

[35] Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Reset-free lifelong learning with skill-space planning. *arXiv preprint arXiv:2012.03548*, 2020.

[36] Eric Mitchell, Rafael Rafailov, Xue Bin Peng, Sergey Levine, and Chelsea Finn. Offline meta-reinforcement learning with advantage weighting. In *International Conference on Machine Learning*, pages 7780–7791. PMLR, 2021.

[37] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. corr abs/1709.10089 (2017). *arXiv preprint arXiv:1709.10089*, 2017.

[38] Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[39] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

[40] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

[41] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[42] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pages 745–750, 2007.

[43] Vitchyr H Pong, Ashvin Nair, Laura Smith, Catherine Huang, and Sergey Levine. Offline meta-reinforcement learning with online self-supervision. *arXiv preprint arXiv:2107.03974*, 2021.

[44] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

[45] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[46] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pages 5331–5340. PMLR, 2019.

[47] Stefan Schaal et al. Learning from demonstration. *Advances in neural information processing systems*, pages 1040–1046, 1997.

[48] Archit Sharma, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Autonomous reinforcement learning via subgoal curricula. *Advances in Neural Information Processing Systems*, 34, 2021.

[49] Archit Sharma, Kelvin Xu, Nikhil Sardana, Abhishek Gupta, Karol Hausman, Sergey Levine, and Chelsea Finn. Autonomous reinforcement learning: Formalism and benchmarking. *arXiv preprint arXiv:2112.09605*, 2021.

[50] Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations. *arXiv preprint arXiv:2102.06177*, 2021.

[51] Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. *arXiv preprint arXiv:1707.04175*, 2017.

[52] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[53] Ziyu Wang, Alexander Novikov, Konrad Zolna, Jost Tobias Springenberg, Scott Reed, Bobak Shahriari, Noah Siegel, Josh Merel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. *arXiv preprint arXiv:2006.15134*, 2020.

[54] Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proceedings of the 24th international conference on Machine learning*, pages 1015–1022, 2007.

[55] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[56] Kelvin Xu, Siddharth Verma, Chelsea Finn, and Sergey Levine. Continual learning of control primitives:

Skill discovery via reset-games. *arXiv preprint arXiv:2011.05286*, 2020.

[57] Zhiyuan Xu, Kun Wu, Zhengping Che, Jian Tang, and Jieping Ye. Knowledge transfer in multi-task deep reinforcement learning for continuous control. *arXiv preprint arXiv:2010.07494*, 2020.

[58] Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. *arXiv preprint arXiv:2003.13661*, 2020.

[59] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782*, 2020.

[60] Tianhe Yu, Aviral Kumar, Yevgen Chebotar, Karol Hausman, Sergey Levine, and Chelsea Finn. Conservative data sharing for multi-task offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[61] Henry Zhu, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey Levine. The ingredients of real-world robotic reinforcement learning. *arXiv preprint arXiv:2004.12570*, 2020.

[62] Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. *arXiv preprint arXiv:1910.08348*, 2019.

Appendix

A. Implementation Details and Hyperparameters

We now provide details of the various hyperparameters used in our experiments.

- **AWAC**: Our hyperparameters for AWAC are listed in Table V.
- **Multi-Task RL**: We use the same hyperparameters as AWAC. We modify the policy and Q function architectures to accept two additional one-hot task indices. These task indices go unused during offline learning, but we use them to label the new data during online training.
- **R3L**: For this method, we keep the same RL hyperparameters as Table V. For the RND networks we use the same CNN architecture as the policy and Q function networks but set the output dimension to 5.
- **Oracle**: For this method, we keep the same RL hyperparameters as Table V, but learn a single-task policy. The offline dataset consists of 512 trajectories.
- **ARIEL**: During online fine-tuning on new tasks with ARIEL, we keep the same hyperparameters as Table V, but use the path lengths listed in Table VI, VII, and VIII. For CEM, we use a Gaussian mixture model as the sampling distributions with a number of components equal to the number of tasks in the prior data. In simulation, we update the sampling distributions every 10 trajectories, fitting them to the $J = 25$ most recent successful task embeddings. In the real world domains, we update the sampling distributions every 10 trajectories, fitting them to the $J = 10$ most recent successful task embeddings.

Hyperparameter	Value
Target Network Update Frequency	1 step
Discount Factor γ	0.9666
Beta	0.01
Batch Size	64
Meta Batch Size	8
Soft Target τ	$5e^{-3}$
Policy Learning Rate	$3e^{-4}$
Q Function Learning Rate	$3e^{-4}$
Reward Scale	1.0
Alpha	0.0
Policy Weight Decay	$1e^{-4}$
Clip Score	0.5

TABLE V
HYPERPARAMETERS FOR AWAC FOR SIMULATED DOMAINS

B. Simulation

We utilize a Pybullet-based simulation [6] containing 3D object models from the Shapenet dataset [4] to test our method on diverse objects. We utilize a near-convex decomposition of the models in order to maintain good contact physics. The following is an example trajectory picking up an cylindrical object and placing it into a container.

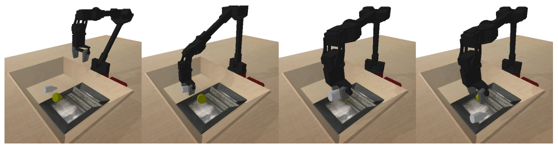

Fig. 7. Example pick and place trajectory in simulation.

C. Simulation Dataset Details

In this section, we provide additional details on the environments and datasets used in our simulation experiments, the results for which were presented in Section VII-A. In Figure 8, we show the set of training and testing objects used in the various pick and place tasks. In Table we provide details on number of tasks, number of episodes and other dataset properties.

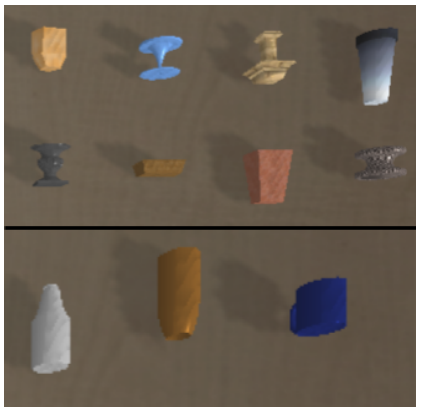

Fig. 8. Simulation training (upper) and test (lower) objects. Our offline prior dataset consists of only training objects.

Attribute	Value
Timesteps per Offline Trajectory	30
Timesteps per Exploration Trajectory	40
Forward Tasks	8
Backward Tasks	8
Number of Trajectories Per Task	512

TABLE VI
SIMULATED TASKS PRIOR DATA DETAILS

D. Real-World Dataset Details

In this section, we provide additional details in Tables VII and VIII on the environments and datasets used in our real world experiments, the results for which were presented in Section VII-B. The real-robot dataset consists of a carefully-selected set of stuffed animals, rigid shapes, and more visually-complex toys. We utilize scripted policies in order to collected a large amount of data interacting with these objects.In the container environment, there two objects in the scene at once to provide better task specification.

Attribute	Value
Timesteps per Offline Trajectory	15
Timesteps per Exploration Trajectory	20
Forward Tasks	20
Backward Tasks	20
Number of Trajectories Per Task	500

TABLE VII
REAL-WORLD PICK AND PLACE PRIOR DATA DETAILS

Attribute	Value
Timesteps per Offline Trajectory	30
Timesteps per Exploration Trajectory	35
Forward Tasks	4
Backward Tasks	4
Number of Trajectories Per Task	150

TABLE VIII
REAL-WORLD DRAWER PRIOR DATA DETAILS

E. Generalization Test Objects

To test the generalization performance of our fine-tuned policy, we utilize 6 different objects depicted in Figure 9. The top 3 objects are used for testing the robustness of the tiger container policy, while the bottom 3 objects are for the drawer task. Note that although the objects chosen to test drawer generalization are seen in the training set for the container task, they are not contained in the offline buffer for the drawer task.

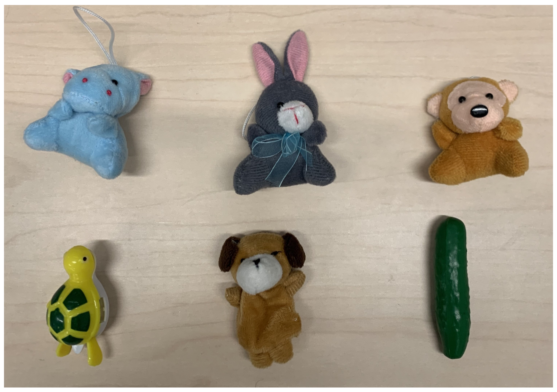

Fig. 9. Generalization objects for tiger (upper) and drawer (lower) tasks.

F. Architecture Details

In Table IX, we provide details of the CNN architecture used for our policy and Q-function networks.

G. Simulation Offline Success Rate Plots

In Figure 10, we provide the learning curves for the offline phase of the **R3L** and **Multi-task RL** baselines.

Attribute	Value
Input Width	48
Input Height	48
Input Channels	3
Kernel Sizes	[3, 3, 3]
Number of Channels	[16, 16, 16]
Strides	[1, 1, 1]
Fully Connected Layers	[1024, 512, 256]
Paddings	[1, 1, 1]
Pool Type	Max 2D
Pool Sizes	[2, 2, 1]
Pool Strides	[2, 2, 1]
Pool Paddings	[0, 0, 0]
Image Augmentation	Yes
Image Augementation Padding	4

TABLE IX
CNN ARCHITECTURE FOR POLICY AND Q-FUNCTION NETWORKS

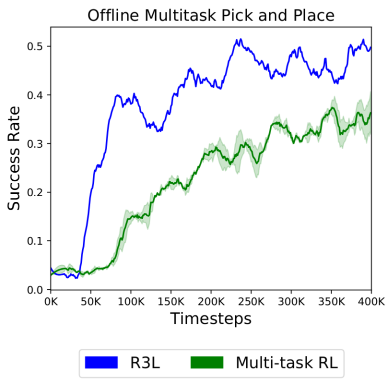

Fig. 10. We see that the multitask pick and place success rates learned by our offline RL baselines show a significant amount of learning, even though they perform poorly on test tasks as seen in Figure 5