Demonstration-Guided Reinforcement Learning with Learned Skills

Karl Pertsch, Youngwoon Lee, Yue Wu, Joseph Lim
University of Southern California
`{pertsch, lee504, ywu73061, limjj}@usc.edu`

Abstract

Demonstration-guided reinforcement learning (RL) is a promising approach for learning complex behaviors by leveraging both reward feedback and a set of target task demonstrations. Prior approaches for demonstration-guided RL treat every new task as an independent learning problem and attempt to follow the provided demonstrations step-by-step, akin to a human trying to imitate a completely unseen behavior by following the demonstrator's exact muscle movements. Naturally, such learning will be slow, but often new behaviors are not completely unseen: they share subtasks with behaviors we have previously learned. In this work, we aim to exploit this shared subtask structure to increase the efficiency of demonstration-guided RL. We first learn a set of reusable skills from large offline datasets of prior experience collected across many tasks. We then propose an algorithm for demonstration-guided RL that efficiently leverages the provided demonstrations by following the demonstrated *skills* instead of the primitive actions, resulting in substantial performance improvements over prior demonstration-guided RL approaches. We validate the effectiveness of our approach on long-horizon maze navigation and complex robot manipulation tasks.

1 Introduction

Humans are remarkably efficient at acquiring new skills from demonstrations: often a single demonstration of the desired behavior and a few trials of the task are sufficient to master it (Bekkering et al., 2000; Al-Abood et al., 2001; Hodges et al., 2007). To allow for such efficient learning, we can leverage a large number of previously learned behaviors (Al-Abood et al., 2001; Hodges et al., 2007). Instead of imitating precisely each of the demonstrated muscle movements, humans can extract the performed *skills* and leverage the rich repertoire of already acquired skills to efficiently reproduce the desired behavior.

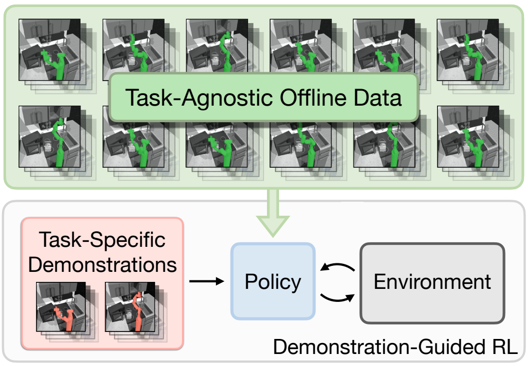

Figure 1: Conventional demonstration-guided reinforcement learning approaches aim to accelerate RL with a small set of task-specific demonstrations by simply imitating the state-action mapping in the demonstrations, which limits their application to relatively short and simple tasks. For efficient learning, we propose to leverage large, task-agnostic datasets collected across many different tasks by (1) acquiring a rich motor skill repertoire from such offline data and (2) understanding and imitating the demonstrations based on the skill repertoire.

Demonstrations are also commonly used in reinforcement learning (RL) to guide exploration and improve sample efficiency (Vecerik et al., 2017; Hester et al., 2018; Rajeswaran et al., 2018; Nair et al., 2018; Zhu et al., 2018). However, such demonstration-guided RL approaches attempt to learn tasks *from scratch*: analogous to a human trying to imitate a completely unseen behavior by following every demonstrated muscle movement, they try to imitate the *primitive actions* performed in the provided demonstrations. As with humans, policies learned with such step-by-step imitation are brittle (Ross et al., 2011), and thus these approaches require a large number of demonstrations and environment interactions to learn a new task.

We propose to improve the efficiency of demonstration-guided RL by leveraging prior experience in the form of an offline "task-agnostic" experience dataset, collected not on one but across many tasks (see Figure 1). Given such a dataset, our approach extracts reusable skills: robust short-horizon behaviors that can be recombined to learn new tasks. Like a human imitating complex behaviors through the chaining of known skills, we can leverage this rich repertoire of skills for efficient demonstration-guided RL on a new task by guiding policy learning using the demonstrated *skills* instead of the primitive actions.

Concretely, we propose a demonstration-guided RL algorithm that builds on recent works in skill-based RL (Pertsch et al., 2020a; Ajay et al., 2020) in order to learn low-dimensional representations of short-horizon skills from offline datasets, and then learns new tasks efficiently by leveraging these skills to follow a given set of demonstrations. Across challenging navigation and robotic manipulation tasks we find that our approach significantly improves the learning efficiency over prior demonstration-guided RL approaches.

In summary, the contributions of our work are threefold: (1) we introduce the problem of leveraging task-agnostic offline datasets for accelerating demonstration-guided RL on unseen tasks, (2) we propose a skill-based algorithm for efficient demonstration-guided RL and (3) we show the effectiveness of our approach on a maze navigation and two complex robotic manipulation tasks.

2 Related Work

Imitation learning. Learning from Demonstration, also known as imitation learning (Argall et al., 2009), is a common approach for learning complex behaviors by leveraging a set of demonstrations. Most prior approaches for imitation learning are either based on behavioral cloning (BC, Pomerleau (1989)), which uses supervised learning to mimic the demonstrated actions, or inverse reinforcement learning (IRL, Abbeel & Ng (2004); Ho & Ermon (2016)), which infers a reward from the demonstrations and then trains a policy to optimize it. However, BC commonly suffers from distribution shift and struggles to learn robust policies (Ross et al., 2011), while IRL's joint optimization of reward and policy can result in unstable training. More generally, the performance of most imitation learning methods is upper bounded by the performance of the demonstrator.

Demonstration-guided RL. A number of prior works aim to mitigate these problems by combining reinforcement learning with imitation learning. This allows the agent to leverage demonstrations for overcoming exploration challenges in RL while using RL to increase robustness and performance of the imitation learning policies. Prior work on demonstration-guided RL can be categorized into three groups: (1) approaches that use BC to initialize and regularize policies during RL training (Rajeswaran et al., 2018; Nair et al., 2018), (2) approaches that place the demonstrations in the replay buffer of an off-policy RL algorithm (Vecerik et al., 2017; Hester et al., 2018), and (3) approaches that augment the environment rewards with rewards extracted from the demonstrations (Zhu et al., 2018; Peng et al., 2018; Merel et al., 2017). While these approaches can leverage demonstrations to improve the efficiency of RL, they all treat each new task as an *independent* learning problem, i.e., attempt to learn policies from scratch without taking any prior experience into account. As a result, they require many demonstrations to learn effectively, which is especially expensive since a new set of task-specific demonstrations needs to be collected for every new task.

Online RL with offline datasets. As an alternative to expensive task-specific demonstrations, multiple recent works have proposed to accelerate reinforcement learning by leveraging *task-agnostic* experience in the form of large datasets collected across many tasks (Pertsch et al., 2020a; Siegel et al., 2020; Nair et al., 2020; Ajay et al., 2020; Singh et al., 2021; 2020). In contrast to demonstrations, such task-agnostic datasets can be collected cheaply from a variety of sources, e.g., using previously trained agents across a diverse set of tasks (Fu et al., 2020; Gulcehre et al., 2020), through agents autonomously exploring their environment (Hausman et al., 2018; Sharma et al., 2020) or via human teleoperation (Schaal et al., 2005; Gupta et al., 2019; Mandlekar et al., 2018; Lynch et al., 2020). Once collected, the dataset can be reused for learning many downstream tasks. Though being cheaper to collect, the task-agnostic data will usually not contain demonstrations for the downstream task, making learning less efficient than demonstration-guided approaches.

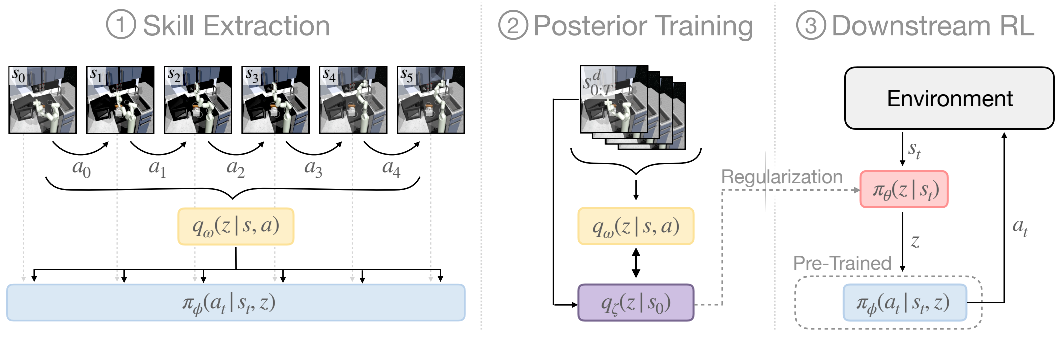

Figure 2: Our approach combines task-agnostic experience and task-specific demonstrations to efficiently learn target tasks in three steps: (1) extract skills from task-agnostic offline data, (2) learn a task-specific skill posterior from demonstrations, and (3) learn a high-level skill policy for the target task using prior knowledge from both task-agnostic offline data and task-specific demonstrations. **Left**: Skill embedding model with skill extractor (**yellow**) and low-level skill policy (**blue**). **Middle**: Training of skill posterior (**purple**) from demonstrations. **Right**: Training of high-level skill policy (**red**) on a downstream task using the pre-trained skill representation and regularization via the skill posterior (skill prior regularization omitted for clarity).

Skill-based RL. One class of approaches for leveraging such offline datasets that is particularly suited for learning long-horizon behaviors is skill-based RL (Hausman et al., 2018; Merel et al., 2019; Kipf et al., 2019; Merel et al., 2020; Shankar et al., 2019; Whitney et al., 2020; Gupta et al., 2019; Lee et al., 2020; Lynch et al., 2020; Pertsch et al., 2020b;a). These methods extract a set of reusable skills from task-agnostic datasets and learn new tasks by recombining them. Yet, such approaches perform *reinforcement learning* over the set of extracted skills to learn the downstream task. Although being more efficient than RL over primitive actions, they still require a large number of environment interactions to learn long-horizon tasks. In our work we combine the best of both worlds: by using large, task-agnostic datasets and a small number of task-specific demonstrations, we accelerate the learning of long-horizon tasks while reducing the number of required demonstrations.

3 Approach

Our goal is to use skills extracted from task-agnostic prior experience data to improve the efficiency of demonstration-guided RL on a new task. We aim to leverage a set of provided demonstrations by following the performed *skills* as opposed to the primitive actions. Therefore, we need a model that can (1) leverage prior data to learn a rich set of skills and (2) identify the skills performed in the demonstrations in order to follow them. In the following, we will formally define our problem setting, briefly summarize relevant prior work on RL with learned skills and then describe our approach for demonstration-guided RL.

3.1 Preliminaries

Problem Formulation We assume access to two types of datasets: a large task-agnostic offline dataset and a small task-specific demonstration dataset. The task-agnostic dataset $\mathcal{D} = \{s_t, a_t, ...\}$ consists of trajectories of meaningful agent behaviors, but includes no demonstrations of the target task. We only assume that its trajectories contain *short-horizon* behaviors that can be reused to solve the target task. Such data can be collected without a particular task in mind using a mix of sources, e.g., via human teleoperation, autonomous exploration, or through policies trained for other tasks. Since it can be used to accelerate *many* downstream task that utilize similar short-term behaviors we call it *task-agnostic*. In contrast, the task-specific data is a much smaller set of demonstration trajectories $\mathcal{D}_{\text{demo}} = \{s^d_t, a^d_t, ...\}$ that are specific to a single target task.

The downstream learning problem is formulated as a Markov decision process (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, R, \rho, \gamma)$ of states, actions, transition probabilities, rewards, initial state distribution, and discount factor. Our goal is to learn a policy $\pi_\theta(a|s)$ with parameters θ that maximizes the discounted sum of rewards $J(\theta) = \mathbb{E}_\pi\big[\sum_{t=0}^{T-1} J_t\big] = \mathbb{E}_\pi\big[\sum_{t=0}^{T-1} \gamma^t r_t\big]$, where T is the episode horizon.

Skill Prior RL Our goal is to extract skills from task-agnostic experience data and reuse them for demonstration-guided RL. Prior work has investigated the reuse of learned skills for accelerating RL (Pertsch et al., 2020a). In this section, we will briefly summarize their proposed approach Skill Prior RL (SPiRL) and then describe how our approach improves upon it in the *demonstration-guided* RL setting.

SPiRL defines a skill as a sequence of H consecutive actions $\boldsymbol{a} = \{a_t, \dots, a_{t+H-1}\}$, where the skill horizon H is a hyperparameter. SPiRL uses the task-agnostic data to jointly learn (1) a generative model of skills $p(\boldsymbol{a}|z)$, that decodes latent skill embeddings z into executable action sequences, and (2) a state-conditioned prior distribution $p(z|s)$ over skill embeddings. For learning a new downstream task, SPiRL trains a high-level skill policy $\pi(z|s)$ whose outputs get decoded into executable actions using the pre-trained skill decoder. Crucially, the learned skill prior is used to guide the policy during downstream RL by maximizing the following divergence-regularized RL objective:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\Bigg[\sum_{t=0}^{T-1} r(s_t, z_t) - \alpha D_{\text{KL}}\big(\pi_\theta(z_t|s_t), p_{\boldsymbol{a}}(z_t|s_t)\big)\Bigg]. \tag{1}$$

Here, the KL-divergence term ensures that the policy remains close to the learned skill prior, guiding exploration during RL. By combining this guided exploration with temporal abstraction via the learned skills, SPiRL substantially improves the efficiency of RL on long-horizon tasks.

3.2 Skill Representation Learning

To effectively reuse skills extracted from task-agnostic data for demonstration-guided RL, we need an expressive skill representation. The open-loop skill decoder used in SPiRL (see Section 3.1) is not conditioned on states and therefore limits the expressiveness of the learned skills. Instead, we follow prior work on skill-based RL (Lynch et al., 2020; Ajay et al., 2020) and extend SPiRL's low-level decoder to a full closed-loop skill policy $\pi(a|s,z)$ that is conditioned on the current environment state. In our experiments we found this closed-loop decoder to improve performance (see Section C for an empirical comparison).

Figure 2 (left) summarizes our skill learning model. It consists of two parts: the skill inference network $q_\omega(z|s_{0:H-1}, a_{0:H-2})$ and the closed-loop skill policy $\pi_\phi(a_t|s_t, z)$. Note that in contrast to SPiRL the skill inference network is state conditioned to account for the state-conditioned low-level policy. During training we randomly sample an H-step state-action trajectory from the task-agnostic dataset. We then pass it to the skill inference network, which predicts the low-dimensional skill embedding z, that encodes the subtask information. This skill embedding is then input into the low-level policy $\pi_\phi(a_t|s_t, z_t)$ for every input state. The policy is trained to imitate the given action sequence, thereby learning to reproduce the behaviors encoded by the skill embedding z.

We optimize the latent skill representation using variational inference, which leads to our full skill learning objective:

$$\max_{\phi,\omega} \mathbb{E}_q\Bigg[\underbrace{\prod_{t=0}^{H-2} \log \pi_\phi(a_t|s_t, z)}_{\text{behavioral cloning}} - \beta\big(\underbrace{\log q_\omega(z|s_{0:H-1}, a_{0:H-2}) - \log p(z)}_{\text{embedding regularization}}\big)\Bigg].$$

We use a unit Gaussian prior $p(z)$ and weight the embedding regularization term with a factor β (Higgins et al., 2017).

3.3 Demonstration-Guided RL with Learned Skills

To leverage the learned skills for accelerating demonstration-guided RL on a new task, we propose to use a hierarchical policy learning scheme (see Figure 2, right): we learn a high-level policy $\pi_\theta(z|s)$ which outputs latent skill embeddings z and transfer the pre-trained low-level policy.

Our goal is to leverage the task-specific demonstrations to guide learning of the high-level policy on the new task. In Section 3.1, we showed how SPiRL (Pertsch et al., 2020a) leverages a learned *skill prior* $p_{\boldsymbol{a}}(z|s)$ to guide exploration. However, this prior is task-agnostic, i.e., it encourages exploration of *all* skills that are meaningful to be explored, independent of which task the agent is trying to solve. Even though SPiRL's objective makes learning with a large number of skills more efficient, it still encourages the policy to explore many skills that are not relevant to the downstream task.

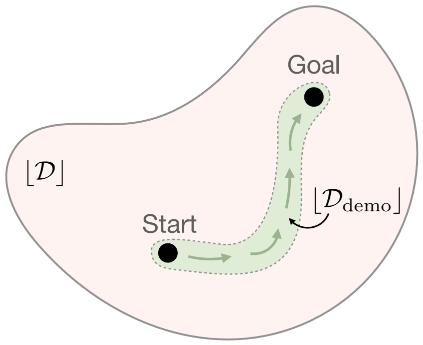
Goal
$\lfloor\mathcal{D}\rfloor$
$\lfloor\mathcal{D}_{\text{demo}}\rfloor$
Start

Figure 3: We leverage prior experience data $\mathcal{D}$ and demonstration data D_{demo}. Our approach is guided by the task-specific skill posterior $q_\zeta(z|s)$ within the support of the demonstrations (**green**) and by the task-agnostic skill prior $p_{\boldsymbol{a}}(z|s)$ otherwise (**red**). The agent also receives a reward bonus for reaching states within the support of the demonstrations.

In this work, we instead propose to leverage target task demonstrations to learn a *task-specific* skill distribution, which we call *skill posterior* $q_\zeta(z|s)$ (in contrast to the skill prior it is conditioned on the target task, hence "posterior"). We train this skill posterior by using the pre-trained skill inference model $q_\omega(z|s_{0:H-1}, a_{0:H-2})$ to extract the embeddings for the skills performed in the demonstration sequences (see Figure 2, middle):

$$\min_\zeta \mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{demo}}} D_{\text{KL}}\big(q_\omega(z|s_{0:H-1}, a_{0:H-2}), q_\zeta(z|s_0)\big), \tag{2}$$

where D_{KL} denotes the Kullback-Leibler divergence.

A naive approach for leveraging the skill posterior is to simply use it to replace the skill prior in Equation 1, i.e., to regularize the policy to stay close to the skill posterior in every state. However, the trained skill posterior is only accurate within the support of the demonstration dataset $\lfloor\mathcal{D}_{\text{demo}}\rfloor$. Since $|\mathcal{D}_{\text{demo}}| \ll |\mathcal{D}|$, this support will necessarily only be a small subset of all states (see Figure 3) and thus the skill posterior will often provide incorrect guidance in states outside the demonstrations' support.

Instead, we propose to use a three-part objective to guide the policy during downstream learning. Our goal in formulating this objective is to (1) follow the skill posterior *within* the support of the demonstrations, (2) follow the skill prior *outside* the demonstration support, and (3) encourage the policy to reach states *within* the demonstration support. Crucial for all three components is to determine whether a given state is within the support of the demonstration data. We propose to use a learned discriminator $D(s)$ to answer this question. It is trained to distinguish demonstration and non-demonstration states using samples from the demonstration and task-agnostic datasets, respectively. Once trained, we use its output to weight terms in our objective that regularize the policy towards the skill prior or posterior. Additionally, we provide a reward bonus for reaching states which the discriminator classifies as being within the demonstration support. This results in the following J_t for our full RL objective:

$$J_t = \tilde{r}(s_t, z_t) - \underbrace{\alpha_q D_{\text{KL}}(\pi_\theta(z_t|s_t), q_\zeta(z_t|s_t)) \cdot D(s_t)}_{\text{posterior regularization}} - \underbrace{\alpha D_{\text{KL}}(\pi_\theta(z_t|s_t), p_{\boldsymbol{a}}(z_t|s_t)) \cdot (1 - D(s_t))}_{\text{prior regularization}},$$

$$\text{with } \tilde{r}(s_t, z_t) = (1-\kappa)\cdot r(s_t, z_t) + \underbrace{\kappa \cdot \big[\log D(s_t) - \log\big(1 - D(s_t)\big)\big]}_{\text{discriminator reward}}.$$

The weighting factor κ is a hyperparameter; α and α_q are either constant or tuned automatically via dual gradient descent (Haarnoja et al., 2018b). For policy optimization, we use a modified version of the SPiRL algorithm (Pertsch et al., 2020a), which itself is based on Soft Actor-Critic (Haarnoja et al., 2018a) (for the full algorithm see appendix, Section A).

The discriminator reward follows common reward formulations used in adversarial imitation learning (Finn et al., 2016; Fu et al., 2018; Zhu et al., 2018; Kostrikov et al., 2019).[1] More generally, our formulation combines IRL-like and BC-like objectives: it uses learned rewards *and* it directly trains the policy to match the demonstration's skill distribution.

[1]We found that using the pre-trained discriminator weights led to stable training, but one could perform full adversarial training by finetuning $D(s)$ with rollouts from the downstream task training. We leave this investigation for future work.

4 Experiments

4.1 Experimental Setup and Comparisons

To evaluate whether our method can efficiently use the task-agnostic data, we compare it to prior demonstration-guided RL approaches on three complex, long-horizon tasks: a 2D maze navigation task, a robotic kitchen manipulation task and a robotic office cleaning task (see Figure 4, left).

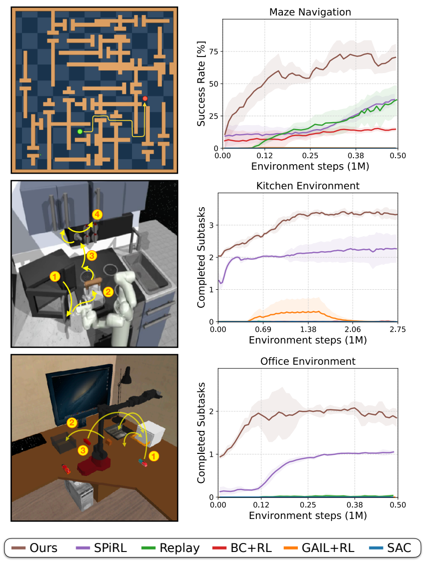

Figure 4: **Left**: Test environments, **top to bottom**: 2D maze navigation, robotic kitchen manipulation and robotic office cleaning. **Right**: Target task performance vs environment steps. By using task-agnostic experience, our approach can more efficiently leverage the provided demonstrations than prior demonstration-guided RL approaches across all tasks. The comparison to SPiRL shows that demonstrations can improve the learning efficiency even if the agent has access to large amounts of prior experience.

Maze Navigation. For this task we adapt the maze navigation task from Pertsch et al. (2020a): we further increase task complexity by adding randomness to the agent's initial position. To solve the task, the agent needs to navigate a maze from the randomized start position to a fixed goal position using planar velocity commands. It only receives a binary reward upon reaching the goal. This environment is challenging for prior demonstration-guided RL approaches since demonstrations of the task span hundreds of time steps, making step-by-step imitation of primitive actions inefficient. We collect a task-agnostic offline experience dataset with 3000 sequences by using a motion planner to find paths between randomly sampled start-goal pairs. For the target task we sample an unseen start-goal pair and then use the same motion planner to collect a much smaller set of 2 demonstrations for reaching the fixed goal position. Although the task-agnostic dataset does not provide demonstrations for reaching the target task goal, it can still be used to extract relevant short-horizon skills like navigating hallways or passing through narrow doors.

Robot Kitchen Environment. We use the environment of Gupta et al. (2019) in which a 7DOF robot agent needs to perform a sequence of four subtasks, such as opening the microwave or switching on the light, in the correct order. It receives a binary reward upon completion of each consecutive subtask. In addition to the long task horizon, this environment requires precise control of a high-DOF manipulator, testing the scalability of demonstration-guided RL approaches. We use 603 teleoperated sequences performing various subtask combinations (from Gupta et al. (2019)) as our task-agnostic experience dataset $\mathcal{D}$ and separate a set of 20 demonstrations for one particular sequence of subtasks, which we define as our target task (see Figure 4, middle). We further mitigate biases in the task-agnostic dataset by resampling trajectories to balance the subtask occurrence probabilities (see Section 4.4 for a detailed discussion of biases in the offline dataset).

Robot Office Environment. In this task, a 5 DOF robot agent needs to clean an office environment by performing subtasks like picking and placing objects or opening and closing a drawer. It receives binary rewards for the completion of each subtask in the correct order. In addition to the challenges of the kitchen environment, this task tests an algorithm's ability to learn long-horizon behaviors with freely manipulatable objects. It also requires the algorithm to handle a greater diversity in the task-agnostic data: we collect 5300 training trajectories by placing the objects at random positions in the environment and performing random subtasks using scripted policies. We also collect a set of 50 demonstrations for the unseen target task with new object locations and an unseen subtask sequence using the same scripted policies.

We compare our approach to multiple prior demonstration-guided RL approaches that represent the different classes of existing algorithms introduced in Section 2. In contrast to our method, these approaches are not able to leverage task-agnostic prior experience:

- **BC + RL**: initializes a policy with behavioral cloning of the demonstrations, then continues to apply BC loss while finetuning the policy with Soft Actor-Critic (SAC, Haarnoja et al. (2018a)), representative of Rajeswaran et al. (2018); Nair et al. (2018).
- **GAIL + RL** (Zhu et al., 2018): combines rewards from the environment and adversarial imitation learning (GAIL, Ho & Ermon (2016)), optimizes the policy using PPO (Schulman et al., 2017).
- **Demo Replay**: initializes the replay buffer of an SAC agent with the demonstration data and applies prioritized replay of the demonstration transitions during updates, representative of Vecerik et al. (2017).

We further compare our approach to RL-only methods to show the benefit of using demonstration data:

- **SAC** (Haarnoja et al., 2018a): is the state-of-the-art model-free RL algorithm, it neither uses offline experience nor demonstrations.
- **SPiRL** (Pertsch et al., 2020a): extracts skills from task-agnostic experience and performs prior-guided RL on the target task (see Section 3.1). We substitute the open-loop skill representation in Pertsch et al. (2020a) with our closed-loop representation (see Section 3.2), since this allows for fairer comparison and we found the performance to be generally superior (for an empirical comparison, see Section C).

For further details on the environments, data collection, and implementation details, see appendix Section B.

4.2 Demonstration-Guided RL with Learned Skills

Maze Navigation. We compare the downstream task performance of the tested methods on the maze navigation task in Figure 4 (right). Prior approaches for demonstration-guided RL struggle to learn the task since task-rewards are sparse and only two demonstrations are provided. With such small coverage, behavioral cloning of the demonstrations' primitive actions leads to brittle policies which are hard to finetune (for an analysis of the influence of the number of demonstrations, see Section 4.3). The Replay agent improves over SAC without demonstrations and partly succeeds at the task, but learning is slow. The IRL-based approach is able to follow part of the demonstrated behavior, but fails to reach the final goal and as a result does not receive the sparse environment reward (see Figure 8 for qualitative results). SPiRL, in contrast, is able to leverage offline, task-agnostic experience to learn to solve the task, but requires a substantial amount of environment interactions: the task-agnostic skill prior encourages the policy to try many skills before converging to the ones that solve the downstream task. In contrast, our approach leverages the task-specific skill posterior to quickly explore the relevant skills, leading to significant efficiency gains.

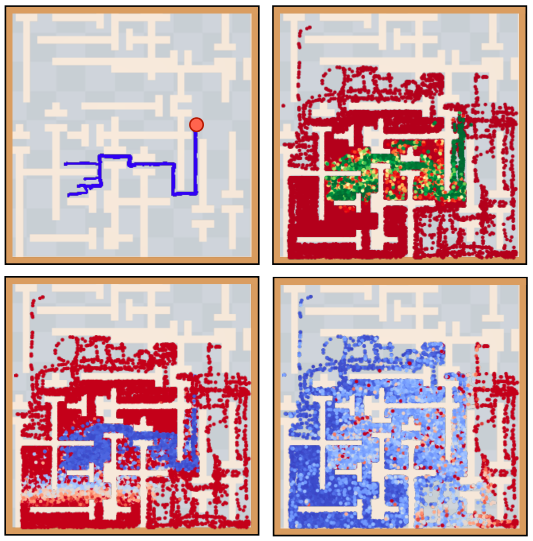

Figure 5: Visualization of our approach on the maze navigation task (with $|\mathcal{D}_{\text{demo}}| = 5$, visualization states collected by rolling out the skill prior). **Top left**: the given demonstration trajectories; **top right**: output of the demonstration discriminator $D(s)$ (the more **green**, the higher the predicted probability of a state to be within demonstration support). **Bottom left**: policy divergences to the skill posterior and **Bottom right**: divergence to the skill prior (**blue** indicates small and **red** high divergence). The discriminator accurately infers the demonstration support, the policy successfully follows the skill posterior only within the demonstration support and the skill prior otherwise.

We further analyze our approach in Figure 5: we visualize the output of the learned discriminator $D(s)$, as well as divergences between the policy and skill prior and posterior. The discriminator accurately estimates the demonstration support, providing a good weighting for prior and posterior regularization, as well as a dense reward bonus. The policy successfully minimizes divergence to the task-specific skill posterior within the demonstration support and follows the skill prior otherwise.

Robotic Manipulation. We show the performance comparison on the robotic manipulation tasks in Figure 4 (right)[2]. Both tasks are more challenging since they require precise control of a high-DOF manipulator. We find that approaches for demonstration-guided RL that do not leverage task-agnostic experience struggle to learn either of the tasks since following the demonstrations step-by-step is inefficient and prone to accumulating errors. SPiRL, in contrast, is able to learn meaningful skills from the offline datasets, but struggles to explore the task-relevant skills and therefore learns slowly. Our approach uses the demonstrations to learn to chain the extracted skills and solve the tasks.

4.3 Ablation Studies

Number of Demonstrations. In Figure 6 (left) we investigate the robustness of demonstration-guided RL approaches to a variety of demonstration set sizes on the maze navigation task. We compare our approach to BC+RL, since we found it to achieve the most consistent results across different numbers of demonstrations. While both approaches benefit from larger demonstration sets, we can observe that the advantage from leveraging prior experience is particularly large in cases with few provided demonstrations. These results are intuitive, since BC+RL needs to use the demonstrations as guidance for learning each low-level action while our approach merely needs to learn to recombine skills it has already mastered using the offline data. Thus, it requires less dense supervision and can cope with fewer demonstrations. Our approach's ability to use task-agnostic data to reduce the number of required demonstrations *and* environment interactions improves two of the most costly aspects of learning tasks in the real world. Thus, the ability to leverage task-agnostic data is an important step for scaling demonstration-guided RL to diverse real-world tasks.

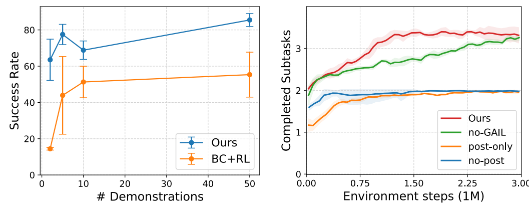

Figure 6: Ablation studies. Performance of our approach for different sizes of the demonstration dataset $|\mathcal{D}_{\text{demo}}|$ on the maze navigation task (**left**) and ablation of different components of our objective on the kitchen manipulation task (**right**).

Objective Ablations. We ablate different parts of our downstream RL objective on the kitchen task in Figure 6 (right). We find that removing the discriminator reward bonus ("*no-GAIL*") leads to slower convergence since the agent lacks a dense reward signal. Naively replacing the skill prior in Equation 1 with the learned skill *posterior* ("*post-only*") fails to learn the task. Without the support-based weighting the policy is constrained to follow the skill posterior in parts of the state space which the posterior was not trained on. Finally, removing the learned skill posterior and optimizing a discriminator bonus augmented reward using SPiRL ("*no-post*") fails to learn the task.

We also ablate the influence of environment rewards to test the efficacy of skill-based learning in the pure imitation setting in appendix, Section E.

4.4 Data Alignment Analysis

We have shown that our approach is able to combine task-agnostic prior experience and task-specific demonstrations to accelerate RL. Since the task-agnostic data can be reused to learn many tasks, the marginal cost of learning a new task is dominated by the cost of collecting the task-specific demonstrations. A key question is therefore: *in what scenarios* does the combination of task-agnostic experience and demonstrations improve efficiency and when can we save costs and solely rely on the task-agnostic dataset?

We hypothesize that the distribution of observed behaviors in the task-agnostic dataset is key to answering this question: if it is not well aligned with the target behaviors, learning is inefficient. Thus, this alignment determines the efficiency gain our method can achieve through the addition

[2]For robot manipulation videos, see https://sites.google.com/view/skill-demo-rl.

of demonstrations. In our previous robotic kitchen experiments we balanced the distribution of behaviors in the task-agnostic dataset to mitigate such biases in the selection of the downstream task. To empirically analyze the effects of good and bad data-task alignment, we now conduct additional experiments with the original, biased data distribution of Gupta et al. (2019). For a detailed analysis of the data biases and the chosen downstream tasks, see Section F. We compare the performance of our method to SPiRL, which solely relies on task-agnostic data.

In the **well-aligned** case (Figure 7, left), we find that both SPiRL and our approach are able to learn the task efficiently. Since the skill prior encourages effective exploration on the downstream task, the benefit of the additional demonstrations leveraged in our method is marginal.

In contrast, if task-agnostic data and downstream task are **mis-aligned** (Figure 7, right), we find that SPiRL struggles to learn the task due to a conflict between the two objectives in Equation 1: maximizing task reward and minimizing divergence from the skill prior. Our approach is able to learn the task more reliably, since it encourages the policy to reach demonstration-like states and then follow the skill posterior, which by design is well-aligned with the downstream task.

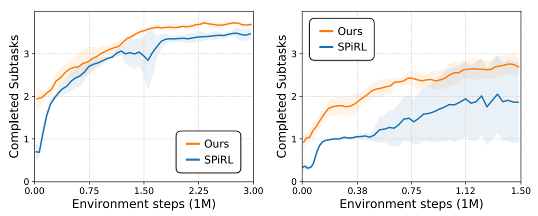

Figure 7: Analysis of offline data-task alignment. The benefit of using demonstrations *in addition* to prior experience diminishes if the prior experience is closely aligned to the target task (**left**). Efficiency gains are high when the learned prior is mis-aligned with the target task, leading to inefficient exploration (**right**).

In summary, our analysis finds that approaches which leverage both task-agnostic data *and* demonstrations, improve over methods that use either of the data sources alone across all tested tasks. We find that combining the data sources is particularly beneficial in two cases:

- **Diverse Task-Agnostic Data.** Demonstrations can focus exploration on the task-relevant skills if the task-agnostic skill prior encourages exploration of a large, potentially irrelevant set of skills (see Section 4.2).
- **Mis-Aligned Task-Agnostic Data.** Demonstrations can compensate mis-alignment between task-agnostic data and target task by guiding exploration with the skill posterior instead of the mis-aligned prior.

Both scenarios are important since it is challenging to control the quality of the prior experience dataset, particularly if collected at scale from multiple sources. Our experiments show that the combination of prior experience and demonstrations can make the learner more robust to fluctuations in the characteristics of the prior experience dataset.

5 Conclusion

We have proposed a novel approach for demonstration-guided RL that is able to leverage task-agnostic experience datasets for accelerated learning of unseen tasks. Our agent extracts a set of reusable skills from the task-agnostic dataset. It then identifies the skills performed in the task-specific demonstrations and uses them to guide downstream RL. In three challenging environments we find that our approach learns unseen tasks more efficiently than both, prior demonstration-guided RL approaches that are not able to leverage task-agnostic experience, as well as skill-based RL methods that cannot effectively incorporate demonstrations.

References

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, 2004.

Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning. *arXiv preprint arXiv:2010.13611*, 2020.

Saleh A Al-Abood, Keith Davids, and Simon J Bennett. Specificity of task constraints and effects of visual demonstrations and verbal instructions in directing learners' search during skill acquisition. *Journal of motor behavior*, 33(3):295–305, 2001.

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

Harold Bekkering, Andreas Wohlschlager, and Merideth Gattis. Imitation of gestures in children is goal-directed. *The Quarterly Journal of Experimental Psychology: Section A*, 53(1):153–164, 2000.

Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *NeurIPS Workshop on Adversarial Training*, 2016.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. In *ICLR*, 2018.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Tom Le Paine, Sergio Gómez Colmenarejo, Konrad Zolna, Rishabh Agarwal, Josh Merel, Daniel Mankowitz, Cosmin Paduraru, et al. Rl unplugged: Benchmarks for offline reinforcement learning. *arXiv preprint arXiv:2006.13888*, 2020.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *CoRL*, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *ICML*, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *ICLR*, 2018.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *AAAI*, 2018.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *NeurIPS*, 2016.

Nicola J Hodges, A Mark Williams, Spencer J Hayes, and Gavin Breslin. What is modelled during observational learning? *Journal of sports sciences*, 25(5):531–545, 2007.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Thomas Kipf, Yujia Li, Hanjun Dai, Vinicius Zambaldi, Edward Grefenstette, Pushmeet Kohli, and Peter Battaglia. Compositional imitation learning: Explaining and executing one task at a time. *ICML*, 2019.

Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *ICLR*, 2019.

Youngwoon Lee, Jingyun Yang, and Joseph J. Lim. Learning to coordinate manipulation skills via skill behavior diversification. In *ICLR*, 2020.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *ICLR*, 2020.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *CoRL*, 2020.

Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, Silvio Savarese, and Li Fei-Fei. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *CoRL*, 2018.

Josh Merel, Yuval Tassa, Dhruva TB, Sriram Srinivasan, Jay Lemmon, Ziyu Wang, Greg Wayne, and Nicolas Heess. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201*, 2017.

Josh Merel, Leonard Hasenclever, Alexandre Galashov, Arun Ahuja, Vu Pham, Greg Wayne, Yee Whye Teh, and Nicolas Heess. Neural probabilistic motor primitives for humanoid control. *ICLR*, 2019.

Josh Merel, Saran Tunyasuvunakool, Arun Ahuja, Yuval Tassa, Leonard Hasenclever, Vu Pham, Tom Erez, Greg Wayne, and Nicolas Heess. Catch & carry: Reusable neural controllers for vision-guided whole-body tasks. *ACM. Trans. Graph.*, 2020.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6292–6299. IEEE, 2018.

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

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.

Karl Pertsch, Youngwoon Lee, and Joseph J. Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on Robot Learning (CoRL)*, 2020a.

Karl Pertsch, Oleh Rybkin, Jingyun Yang, Shenghao Zhou, Kosta Derpanis, Joseph Lim, Kostas Daniilidis, and Andrew Jaegle. Keyframing the future: Keyframe discovery for visual prediction and planning. *L4DC*, 2020b.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Proceedings of Neural Information Processing Systems (NeurIPS)*, pp. 305–313, 1989.

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. In *Robotics: Science and Systems*, 2018.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 627–635, 2011.

Stefan Schaal, Jan Peters, Jun Nakanishi, and Auke Ijspeert. Learning movement primitives. In Paolo Dario and Raja Chatila (eds.), *Robotics Research*. Springer Berlin Heidelberg, 2005.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Tanmay Shankar, Shubham Tulsiani, Lerrel Pinto, and Abhinav Gupta. Discovering motor programs by recomposing demonstrations. In *ICLR*, 2019.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *ICLR*, 2020.

Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *ICLR*, 2020.

Avi Singh, Albert Yu, Jonathan Yang, Jesse Zhang, Aviral Kumar, and Sergey Levine. Cog: Connecting new skills to past experience with offline reinforcement learning. *CoRL*, 2020.

Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. *ICLR*, 2021.

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.

William Whitney, Rajat Agarwal, Kyunghyun Cho, and Abhinav Gupta. Dynamics-aware embeddings. *ICLR*, 2020.

Yuke Zhu, Ziyu Wang, Josh Merel, Andrei Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, János Kramár, Raia Hadsell, Nando de Freitas, and Nicolas Heess. Reinforcement and imitation learning for diverse visuomotor skills. In *Robotics: Science and Systems*, 2018.

Algorithm 1 Demonstration-Guided RL with Learned Skills

```
1: Inputs: $H$-step reward function $\tilde{r}(s_t, z_t)$, reward weight $\gamma$, discount $\eta$, target divergences $\delta, \delta_q$,
   learning rates $\lambda_\pi, \lambda_Q, \lambda_\alpha$, target update rate $\tau$.
2: Initialize replay buffer $\mathcal{D}$, high-level policy $\pi_\theta(z_t|s_t)$, critic $Q_\phi(s_t, z_t)$, target network $Q_{\bar{\phi}}(s_t, z_t)$
3: for each iteration do
4:     for every $H$ environment steps do
5:         $z_t \sim \pi(z_t|s_t)$                                        ▷ sample skill from policy
6:         $s_{t'} \sim p(s_{t+H}|s_t, z_t)$                              ▷ execute skill in environment
7:         $\mathcal{D} \leftarrow \mathcal{D} \cup \{s_t, z_t, \tilde{r}(s_t, z_t), s_{t'}\}$        ▷ store transition in replay buffer
8:     for each gradient step do
9:         $r_\Sigma = (1-\gamma) \cdot \tilde{r}(s_t, z_t) + \gamma \cdot \big[\log D(s_t) - \log\big(1 - D(s_t)\big)\big]$   ▷ compute combined reward
10:        $\bar{Q} = r_\Sigma + \eta\big[Q_{\bar{\phi}}(s_{t'}, \pi_\theta(z_{t'}|s_{t'})) - \big[\alpha_q D_{\mathrm{KL}}\big(\pi_\theta(z_{t'}|s_{t'}), q_\zeta(z_{t'}|s_{t'})\big) \cdot D(s_{t'})$
11:            $+ \alpha D_{\mathrm{KL}}\big(\pi_\theta(z_{t'}|s_{t'}), p_{\boldsymbol{a}}(z_{t'}|s_{t'})\big) \cdot \big(1 - D(s_{t'})\big)\big]$   ▷ compute Q-target
12:        $\theta \leftarrow \theta - \lambda_\pi \nabla_\theta\big[Q_\phi(s_t, \pi_\theta(z_t|s_t)) - \big[\alpha_q D_{\mathrm{KL}}\big(\pi_\theta(z_t|s_t), q_\zeta(z_t|s_t)\big) \cdot D(s_t)$
13:            $+ \alpha D_{\mathrm{KL}}\big(\pi_\theta(z_t|s_t), p_{\boldsymbol{a}}(z_t|s_t)\big) \cdot \big(1 - D(s_t)\big)\big]$   ▷ update policy weights
14:        $\phi \leftarrow \phi - \lambda_Q \nabla_\phi\big[\frac{1}{2}\big(Q_\phi(s_t, z_t) - \bar{Q}\big)^2\big]$          ▷ update critic weights
15:        $\alpha \leftarrow \alpha - \lambda_\alpha \nabla_\alpha\big[\alpha \cdot \big(D_{\mathrm{KL}}(\pi_\theta(z_t|s_t), p_{\boldsymbol{a}}(z_t|s_t)) - \delta\big)\big]$   ▷ update alpha
16:        $\alpha_q \leftarrow \alpha_q - \lambda_\alpha \nabla_{\alpha_q}\big[\alpha_q \cdot \big(D_{\mathrm{KL}}(\pi_\theta(z_t|s_t), q_\zeta(z_t|s_t)) - \delta_q\big)\big]$   ▷ update alpha-q
17:        $\bar{\phi} \leftarrow \tau\phi + (1-\tau)\bar{\phi}$            ▷ update target network weights
18: return trained policy $\pi_\theta(z_t|s_t)$
```

A Full Algorithm

We detail our full algorithm for demonstration-guided RL with learned skills in Algorithm 1. It is based on the SPiRL algorithm for RL with learned skills Pertsch et al. (2020a) which in turn builds on Soft-Actor Critic Haarnoja et al. (2018a), an off-policy model-free RL algorithm. We mark changes of our algorithm with respect to SPiRL and SAC in red in Algorithm 1.

The hyperparameters α and α_q can either be constant, or they can be automatically tuned using dual gradient descent Haarnoja et al. (2018b); Pertsch et al. (2020a). In the latter case, we need to define a set of *target divergences* δ, δ_q. The parameters α and α_q are then optimized to ensure that the expected divergence between policy and skill prior and posterior distributions is equal to the chosen target divergence (see Algorithm 1).

B Implementation and Experimental Details

B.1 Implementation Details: Pre-Training

We introduce our objective for learning the skill inference network $q_\omega(z|s,a)$ and low-level skill policy $\pi_\phi(a_t|s_t, z)$ in Section 3.2. In practice, we instantiate all model components with deep neural networks Q_ω, Π_ϕ respectively, and optimize the full model using back-propagation. We also jointly train our skill prior network $P_{\boldsymbol{a}}$. We follow the common assumption of Gaussian, unit-variance output distributions for low-level policy actions, leading to the following network loss:

$$\begin{aligned}\mathcal{L} = \prod_{t=0}^{H-2} &\big\|a_t - \Pi_\phi(s_t, s_g)\big\|^2 \\ &+ \beta D_{\mathrm{KL}}\big(Q_\omega(s_{0:H-1}, a_{0:H-2}) \;||\; \mathcal{N}(0, I)\big) \\ &+ D_{\mathrm{KL}}\big(\lfloor Q_\omega(s_{0:H-1}, a_{0:H-2})\rfloor \;||\; P_{\boldsymbol{a}}(s_0)\big).\end{aligned}$$

Here $\lfloor\cdot\rfloor$ indicates that we stop gradient flow from the prior training objective into the skill inference network for improved training stability. After training the skill inference network with above objective, we train the skill posterior network Q_ζ by minimizing KL divergence to the skill inference network's output on trajectories sampled from the demonstration data. We minimize the following objective:

$$\mathcal{L}_{\text{post}} = D_{\mathrm{KL}}\big(\lfloor Q_\omega(s_{0:H-1}, a_{0:H-2})\rfloor \;||\; Q_\zeta(s_0)\big)$$

We use a 1-layer LSTM with 128 hidden units for the inference network and 3-layer MLPs with 128 hidden units in each layer for the low-level policy. We encode skills of horizon 10 into 10-dimensional skill representations z. Skill prior and posterior networks are implemented as 5-layer MLPs with 128 hidden units per layer. They both parametrize mean and standard deviation of Gaussian output distributions. All networks use batch normalization after every layer and leaky ReLU activation functions. We tune the regularization weight β to be $1\mathrm{e}{-2}$ for the maze and $5\mathrm{e}{-4}$ for kitchen and office environment.

For the demonstration discriminator $D(s)$ we use a 3-layer MLP with only 32 hidden units per layer to avoid overfitting. It uses a sigmoid activation function on the final layer and leaky ReLU activations otherwise. We train the discriminator with binary cross-entropy loss on samples from task-agnostic and demonstration datasets:

$$\mathcal{L}_{\text{D}} = -\frac{1}{N} \cdot \Big[\underbrace{\sum_{i=1}^{N/2} \log D(s_i^d)}_{\text{demonstrations}} + \underbrace{\sum_{j=1}^{N/2} \log \big(1 - D(s_j)\big)}_{\text{task-agnostic data}} \Big]$$

We optimize all networks using the RAdam optimizer Liu et al. (2020) with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$, batch size 128 and learning rate $1\mathrm{e}{-3}$. On a single NVIDIA Titan X GPU we can train the skill representation and skill prior in approximately 5 hours, the skill posterior in approximately 3 hours and the discriminator in approximately 3 hours.

B.2 Implementation Details: Downstream RL

The architecture of the policy mirrors the one of the skill prior and posterior networks. The critic is a simple 2-layer MLP with 256 hidden units per layer. The policy outputs the parameters of a Gaussian action distribution while the critic outputs a single Q-value estimate. We initialize the policy with the weights of the skill posterior network.

We use the hyperparameters of the standard SAC implementation Haarnoja et al. (2018a) with batch size 256, replay buffer capacity of $1\mathrm{e}6$ and discount factor $\gamma = 0.99$. We collect 5000 warmup rollout steps to initialize the replay buffer before training. We use the Adam optimizer Kingma & Ba (2015) with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and learning rate $3\mathrm{e}{-4}$ for updating policy, critic and temperatures α and α_q. Analogous to SAC, we train two separate critic networks and compute the Q-value as the minimum over both estimates to stabilize training. The corresponding target networks get updated at a rate of $\tau = 5\mathrm{e}{-3}$. The policy's actions are limited in the range $[-2, 2]$ by a $\tanh$ "squashing function" (see Haarnoja et al. (2018a), appendix C).

We use automatic tuning of α and α_q in the maze and the office environment and set both target divergences to 1 and 7 respectively. In the kitchen environment we obtained best results by using constant values of $\alpha = \alpha_q = 5\mathrm{e}{-2}$. In all experiments we set $\kappa = 0.9$.

For all RL results we average the results of three independently seeded runs and display mean and standard deviation across seeds.

B.3 Implementation Details: Comparisons

BC+RL. This comparison is representative of demonstration-guided RL approaches that use BC objectives to initialize and regularize the policy during RL Rajeswaran et al. (2018); Nair et al. (2018). We pre-train a BC policy on the demonstration dataset and use it to initialize the RL policy. We use SAC to train the policy on the target task. Similar to Nair et al. (2018) we augment the policy update with a regularization term that minimizes the L2 loss between the predicted mean of the policy's output distribution and the output of the BC pre-trained policy[3].

[3]We also tried sampling action targets directly from the demonstration replay buffer, but found using a BC policy as target more effective on the tested tasks.

Demo Replay. This comparison is representative of approaches that initialize the replay buffer of an off-policy RL agent with demonstration transitions Vecerik et al. (2017); Hester et al. (2018). In practice we use SAC and initialize a second replay buffer with the demonstration transitions. Since the demonstrations do not come with reward, we heuristically set the reward of each demonstration trajectory to be a high value (100 for the maze, 4 for the robotic environments) on the final transition and zero everywhere else. During each SAC update, we sample half of the training mini-batch from the normal SAC replay buffer and half from the demonstration replay buffer. All other aspects of SAC remain unchanged.

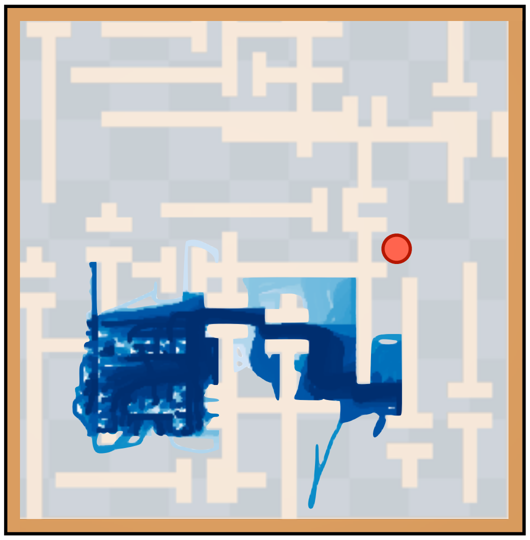

Figure 8: Qualitative results for GAIL+RL on the 2D maze navigation task. The approach is able to leverage the demonstrations to make progress towards the goal (**red**), but since it fails to reach the goal it never obtains the sparse environment reward feedback and as a result fails to solve the full target task.

B.4 Environment Details

Maze Navigation. We adapt the maze navigation task from Pertsch et al. (2020a) which extends the maze navigation tasks from the D4RL benchmark Fu et al. (2020). The starting position is sampled uniformly from a start region and the agent receives a one-time sparse reward of 100 when reaching the fixed goal position, which also ends the episode. The 4D observation space contains 2D position and velocity of the agent. The agent is controlled via 2D velocity commands.

Robot Kitchen Environment. We use the kitchen environment from Gupta et al. (2019). For solving the target task, the agent needs to execute a fixed sequence of four subtasks by controlling an Emika Franka Panda 7-DOF robot via joint velocity and continuous gripper actuation commands. The 30-dimensional state space contains the robot's joint angles as well as object-specific features that characterize the position of each of the manipulatable objects. We mitigate biases in the distribution over subtasks transitions by re-sampling sequences of the provided dataset (see Section F for details). Further, we collect a version of the demonstration dataset with increased support by initializing the environment state to states randomly sampled from the demonstrations and rolling out a random policy for 10 steps. We use this data augmentation technique for increasing the support while pre-training the demonstration discriminator $D(s)$[4].

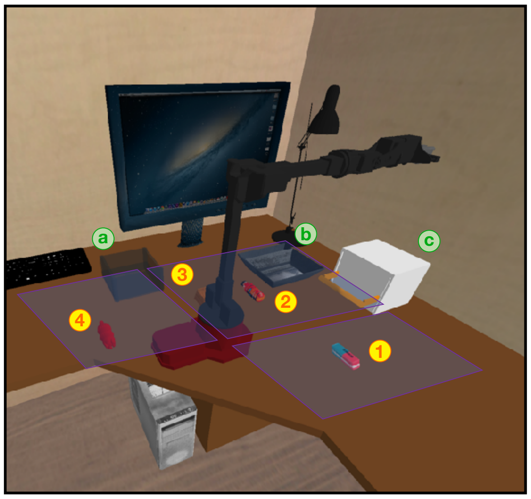

Figure 9: Office cleanup task. The robot agent needs to place a subset of three randomly sampled objects **(1-4)** inside randomly sampled containers **(a-c)**. During task-agnostic data collection we randomly sample the object's initial positions in the shaded areas.

Robot Office Environment. We create a novel office cleanup task in which a 5-DOF WidowX robot needs to place a number of objects into designated containers, requiring the execution of a sequence of pick, place and drawer open and close subtasks (see Figure 9). The agent

[4]We also tried training the GAIL and GAIL+RL baselines with the augmented demonstrations but did not observe any benefit, we report the best results for the baseline in both cases.

controls position and orientation of the end-effector and a continuous gripper actuation, resulting in a 7-dimensional action space. For simulating the environment we build on the Roboverse framework Singh et al. (2020). During collection of the task-agnostic data we randomly sample a subset of three of the four objects as well as a random order of target containers and use scripted policies to execute the task (see Figure 14). We only save successful executions. For the target task we fix object positions and require the agent to place three objects in fixed target containers. The 76-dimensional state space contains the agent's end-effector position and orientation as well as position and orientation of all objects and containers.

C Skill Representation Comparison

In Section 3.2 we described our skill representation based on a closed-loop low-level policy as a more powerful alternative to the action trajectory-based representation of Pertsch et al. (2020a). To compare the performance of the two representations we perform rollouts with the learned skill prior: we sample a skill from the prior and rollout the low-level policy for H steps. We repeat this until the episode terminates and visualize the results for multiple episodes in maze and kitchen environment in Figure 10.

In Figure 10 (top) we see that both representations lead to effective exploration in the maze environment. Since the 2D maze navigation task does not require control in high-dimensional action spaces, both skill representations are sufficient to accurately reproduce behaviors observed in the task-agnostic training data.

In contrast, the results on the kitchen environment (Figure 10, bottom) show that the closed-loop skill representation is able to more accurately control the high-DOF robotic manipulator and reliably solve multiple subtasks per rollout episode.[5] We hypothesize that the closed-loop skill policy is able to learn more robust skills from the task-agnostic training data, particularly in high-dimensional control problems.

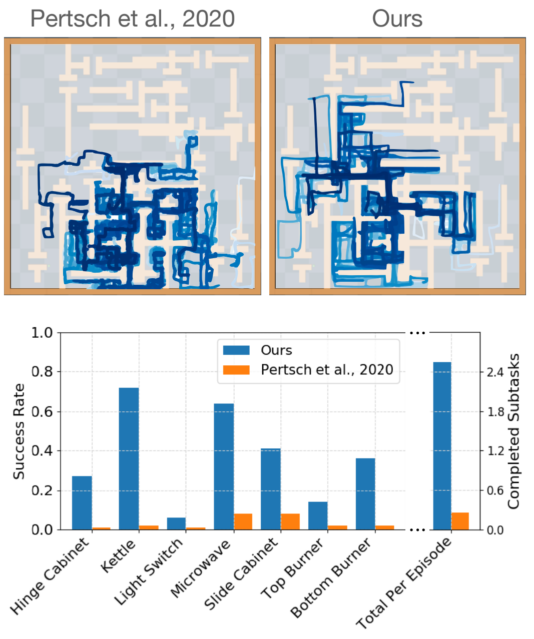

Figure 10: Comparison of our closed-loop skill representation with the trajectory-based representation of Pertsch et al. (2020a). **Top**: Skill prior rollouts for 100 k steps in the maze environment. Both skill representations explore the maze widely. **Bottom**: Subtask success rates for prior rollouts in the kitchen environment. Only the closed-loop skill representation is able to solve multiple subtasks per episode.

D Demonstration-Guided RL Comparisons with Task-Agnostic Experience

In Section 4.2 we compared our approach to prior demonstration-guided RL approaches which are not designed to leverage task-agnostic datasets. We applied these prior works in the setting they were designed for: using only task-specific demonstrations of the target task. Here, we conduct experiments in which we run these prior works using the *combined* task-agnostic and task-specific datasets to give them access to the same data that our approach used.

From the results in Figure 11 we can see that none of the prior works is able to effectively leverage the additional task-agnostic data. In many cases the performance of the approaches is worse than when only using task-specific data (see Figure 4). Since prior approaches are not designed to leverage task-agnostic data, applying them in the combined-data setting can hurt learning on the

[5]See https://sites.google.com/view/skill-demo-rl for skill prior rollout videos with both skill representations in the kitchen environment.

target task. In contrast, our approach can effectively leverage the task-agnostic data for accelerating demonstration-guided RL.

E Skill-Based Imitation Learning

We ablate the influence of the environment reward feedback on the performance of our approach by setting the reward weight $\gamma = 1.0$, thus relying solely on the learned discriminator reward. Our goal is to test whether our approach is able to leverage task-agnostic experience to improve the performance of pure *imitation learning*, i.e., learning to follow demonstrations without environment reward feedback.

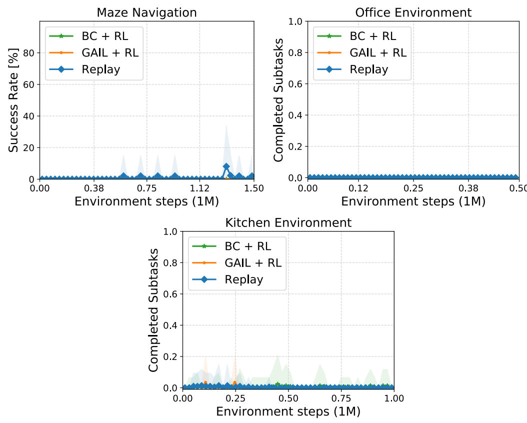

Figure 11: Downstream task performance for prior demonstration-guided RL approaches with combined task-agnostic and task-specific data. All prior approaches are unable to leverage the task-agnostic data, showing a performance decrease when attempting to use it.

We compare our method to common approaches for imitation learning: behavioral cloning (BC, Pomerleau (1989)) and generative adversarial imitation learning (GAIL, Ho & Ermon (2016)). We also experiment with a version of our skill-based imitation learning approach that performs online finetuning of the pre-trained discriminator $D(s)$ using data collected during training of the imitation policy.

We summarize the results of the imitation learning experiments in Figure 12. Learning purely by imitating the demonstrations, without additional reward feedback, is generally slower than demonstration-guided RL. Yet, we find that our approach is able to leverage task-agnostic data to effectively imitate complex, long-horizon behaviors while prior imitation learning approaches struggle. Further, online finetuning of the learned discriminator slightly improves imitation learning performance.

F Kitchen Data Analysis

For the kitchen manipulation experiments we use the dataset provided by Gupta et al. (2019) as task-agnostic pre-training data. It consists of 603 teleoperated sequences, each of which shows the completion of four consecutive subtasks. In total there are seven possible subtasks: opening the microwave, moving the kettle, turning on top and bottom burner, flipping the light switch and opening a slide and a hinge cabinet.

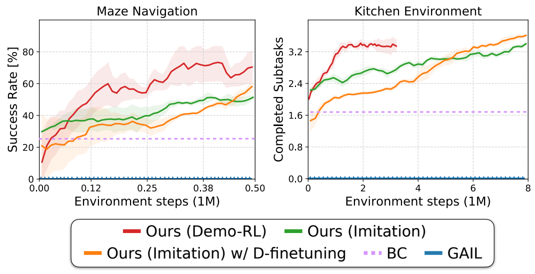

Figure 12: Imitation learning performance on maze navigation and kitchen tasks. Compared to prior imitation learning methods, our approach can leverage prior experience to enable the imitation of complex, long-horizon behaviors. Finetuning the pre-trained discriminator $D(s)$ further improves performance.

In Figure 13 we analyze the transition probabilities between subtasks in the task-agnostic dataset. We can see that these transition probabilities are not uniformly distributed, but instead certain transitions are more likely than others, e.g., it is much more likely to sample a training trajectory in which the agent first opens the microwave than one in which it starts by turning on the bottom burner.

In Section 4.4 we test the effect this bias in transition probabilities has on the learning of target tasks. Concretely, we investigate two cases: good alignment between task-agnostic data and target task and

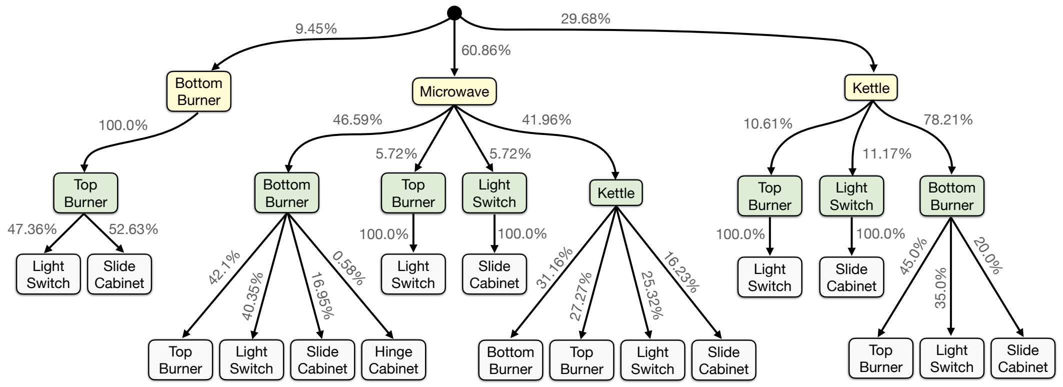

Figure 13: Subtask transition probabilities in the kitchen environment's task-agnostic training dataset from Gupta et al. (2019). Each dataset trajectory consists of four consecutive subtasks, of which we display three (**yellow**: first, **green**: second, **grey**: third subtask). The transition probability to the fourth subtask is always near 100 %. In Section 4.4 we test our approach on a target task with good alignment to the task-agnostic data (*Microwave - Kettle - Light Switch - Hinge Cabinet*) and a target task which is mis-aligned to the data (*Microwave - Light Switch - Slide Cabinet - Hinge Cabinet*).

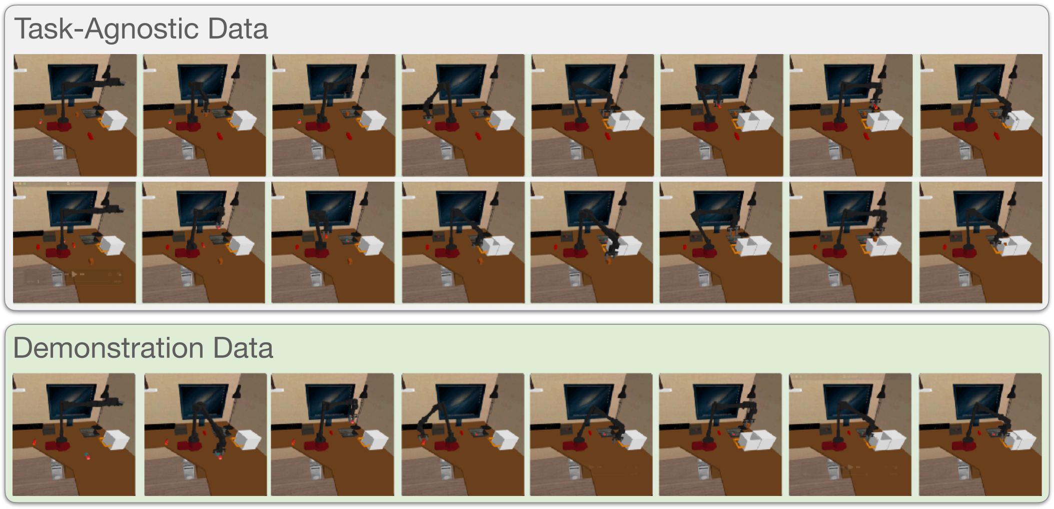

Figure 14: Office cleanup task data collection. The top two sequences show subsampled trajectories from the task-agnostic dataset, the bottom sequence is a demonstration of the target task.

mis-alignment between the two. In the former case we choose the target task *Microwave - Kettle - Light Switch - Hinge Cabinet*, since the required subtask transitions are likely under the training data distribution. For the mis-aligned case we choose *Microwave - Light Switch - Slide Cabinet - Hinge Cabinet* as target task, since particularly the transition from opening the microwave to flipping the light switch is very unlikely to be observed in the training data.

In our experiments in Section 4.2 we minimize bias introduced through the target task choice by balancing the subtask transition probabilities in the task-agnostic dataset. We achieve this by resampling training trajectories such that it is equally likely to sample any of the first two subtask transitions, i.e., after resampling it is equally likely to sample a training trajectory that starts with *Bottom Burner - Top Burner* as it is to sample a trajectory starting with *Microwave - Kettle*. As a result, the task-agnostic data has equal alignment to a large set of possible downstream tasks.