POLICY EXPANSION FOR BRIDGING OFFLINE-TO-ONLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Pre-training with offline data and online fine-tuning using reinforcement learning is a promising strategy for learning control policies by leveraging the best of both worlds in terms of sample efficiency and performance. One natural approach is to initialize the policy for online learning with the one trained offline. In this work, we introduce a policy expansion scheme for this task. After learning the offline policy, we use it as one candidate policy in a policy set. We then expand the policy set with another policy which will be responsible for further learning. The two policies will be composed in an adaptive manner for interacting with the environment. With this approach, the policy previously learned offline is fully retained during online learning, thus mitigating the potential issues such as destroying the useful behaviors of the offline policy in the initial stage of online learning while allowing the offline policy participate in the exploration naturally in an adaptive manner. Moreover, new useful behaviors can potentially be captured by the newly added policy through learning. Experiments are conducted on a number of tasks and the results demonstrate the effectiveness of the proposed approach.

1 INTRODUCTION

Reinforcement learning (RL) has shown great potential in various fields, reaching or even surpassing human-level performances on many tasks (e.g. Mnih et al., 2015; Silver et al., 2017; Schrittwieser et al., 2019; Tsividis et al., 2021). However, since the policy is learned from scratch for a given task in the standard setting, the number of samples required by RL for successfully solving a task is usually large, which limits its applicability in many practical scenarios such as robotics, where physical interaction and data collection has a non-trivial cost.

In many cases, there is a good amount of offline data that has already been available (Kober et al., 2013; Rastgoftar et al., 2018; Cabi et al., 2019), e.g., collected during previous iterations of experiments or from human (e.g. for the task of driving). Instead of ab initio learning as in the common RL setting, how to effectively leverage the already available offline data for helping with online policy learning is an interesting and open problem (Vecerík et al., 2017; Hester et al., 2018; Nair et al., 2018).

Offline RL is an active recent direction that aims to learn a policy by purely using the offline data, without any further online interactions (Fujimoto et al., 2019; Kumar et al., 2020; Fujimoto & Gu, 2021; Levine et al., 2020; Ghosh et al., 2022; Chen et al., 2021; Janner et al., 2021; Yang et al., 2021; Lu et al., 2022; Zheng et al., 2022). It holds the promise of learning from suboptimal data and improving over the behavior policy that generates the dataset (Kumar et al., 2022), but its performance could still be limited because of its full reliance on the provided offline data.

To benefit from further online learning, one possible way is to pre-train with offline RL, and warm start the policy of an online RL algorithm to help with learning and exploration when learning online. While this pre-training + fine-tuning paradigm is natural and intuitive, and has received great success in many fields like computer vision (Ge & Yu, 2017; Kornblith et al., 2019) and natural language processing (Devlin et al., 2018; Radford & Narasimhan, 2018; Brown et al., 2020), it is less widely used in RL. Many early attempts in RL community report a number of negative results along this direction. For example, it has been observed that initializing the policy with offline pre-training and then fine-tuning the policy with standard online RL algorithms (e.g. SAC (Haarnoja et al., 2018)) sometimes suffers from non-recoverable performance drop under certain settings (Nair et al., 2020;

Uchendu et al., 2022), potentially due to the distribution shift between offline and online stages and the change of learning dynamics because of the algorithmic switch.

Another possible way is to use the same offline RL algorithm for online learning. However, it has been observed that standard offline RL methods generally are not effective in fine-tuning with online data, due to reasons such as conservativeness of the method (Nair et al., 2020). Some recent works in offline RL also start to focus on the offline-pre-training + online fine-tuning paradigm (Nair et al., 2020; Kostrikov et al., 2022). For this purpose, they share the common philosophy of designing an RL algorithm that is suitable for both offline and online phases. Because of the unified algorithm across phases, the network parameters (including those for both critics and actor) trained in the offline phase can be reused for further learning in the online phase.

Our work shares the same objective of designing effective offline-to-online training schemes. However, we take a different perspective by focusing on how to bridge offline-online learning, and not on developing yet another offline or online RL method, which is orthogonal to the focus of this work. We will illustrate the idea concretely by instantiating our proposed scheme by applying it on existing RL algorithms (Kostrikov et al., 2022; Haarnoja et al., 2018). The contributions of this work are:

- we highlight the value of *properly connecting* existing offline and online RL methods in order to enjoy the best of both worlds, a perspective that is alternative and orthogonal to developing completely new RL algorithms;
- we propose a simple scheme termed as *policy expansion* for bridging offline and online reinforcement learning. The proposed approach is not only able to preserve the behavior learned in the offline stage, but can also leverage it adaptively during online exploration and along the process of learning;
- we verify the effectiveness of the proposed approach by conducting extensive experiments on various tasks and settings, with comparison to a a number of baseline methods.

2 PRELIMINARIES

We briefly review some related basics in this section, first on model-free RL for online policy learning, and then on policy learning from offline dataset.

2.1 ONLINE REINFORCEMENT LEARNING

Standard model-free RL methods learn a policy that maps the current state s to a distribution of action a as $\pi(s)$. The policy is typically modeled with a neural network $\pi_\theta(s)$ with θ denoting the learnable parameters. To train this policy, there are different approaches including on-policy (Sutton et al., 2000; Schulman et al., 2017) and off-policy RL methods (Lillicrap et al., 2016; Haarnoja et al., 2018; Fujimoto et al., 2018). In this work, we mainly focus on off-policy RL for online learning because of its higher sample efficiency. Standard off-policy RL methods rely on the state-action value function $Q(s, a)$ using TD-learning:

$$Q(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim T(s,a), a' \sim \pi_\theta(s')} \big[Q(s', a') \big],$$

where $T(s, a)$ denotes the dynamics function and $r(s, a)$ the reward. $\gamma \in (0, 1)$ is a discount factor. By definition, $Q(s, a)$ represents the accumulated discounted future reward starting from s, taking action a, and then following policy π_θ thereafter.

The optimization of θ is achieved by maximizing the following function:

$$\max_\theta \mathbb{E}_{s \sim \mathcal{D}} \mathbb{E}_{a \sim \pi_\theta} Q(s, a). \tag{1}$$

where $\mathcal{D}$ denotes replay buffer for storing online trajectories. In the typical RL setting, learning is conducted from scratch by initializing all parameters randomly and interacting with the world with the randomly initialized policy. In practive, $Q(s, a)$ is implemented as a neural network $Q_\phi(s, a)$.

2.2 POLICY LEARNING FROM OFFLINE DATASET

Policy learning from offline datasets has been investigated from different perspectives. Given expert-level demonstration data, behavior cloning (BC) (Pomerleau, 1988; Bain & Sammut, 1996) is an

effective approach for offline policy learning because of its simplicity and effectiveness. In fact, in some recent work, BC has shown to perform competitively with some offline RL methods (Fujimoto & Gu, 2021; Chen et al., 2021). Given a dataset $\mathcal{D}_{\text{offline}} = \{(s_i, a_i)\}$ consisting of expert's state action pairs (s_i, a_i), BC trains the policy with maximum likelihood over the data:

$$\max_{\theta} \mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{offline}}} \log \pi_\theta(a|s). \tag{2}$$

Although BC has the benefits of reducing the policy learning task to an ordinary supervised learning task, it suffers from the well-known distributional shift issue (Codevilla et al., 2019; Muller et al., 2005; de Haan et al., 2019; Wen et al., 2020). Another limitation is that BC has a relatively strong requirements on the data quality, and is not good at learning from suboptimal data.

Offline RL is a category of methods that are more suitable for policy learning from noisy and suboptimal offline data (Kumar et al., 2022). When focusing on offline learning only, the core challenge is how to address the extrapolation error due to querying the critic function with out-of-distribution actions (Fujimoto et al., 2019; Kumar et al., 2020). Common strategies include constraining the actions to be close to dataset actions (Fujimoto et al., 2019; Fujimoto & Gu, 2021), and constraining the critic to be conservative for out of data distribution actions (Kumar et al., 2020). The recent implicit Q-learning (IQL) method (Kostrikov et al., 2022) addresses this issue by learning a value network to match the expectile of the critic network, thus avoiding querying the critic with the actions not in the offline dataset. For policy update, IQL use a weighted BC formulation

$$\max_{\theta} \mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{offline}}} w(s,a) \cdot \log \pi_\theta(a|s), \tag{3}$$

where $w(s, a)$ denotes a data dependent weight, typically calculated based on the estimated advantages (Nair et al., 2020; Xu et al., 2022; Kostrikov et al., 2022).

3 Offline and Online RL Revisited: Connections and Gaps

Connections between Offline and Online RL. Offline RL and online RL are closely connected in many aspects. Historically, many offline RL approaches are branched off from off-policy RL algorithms to the full offline setting (Fujimoto et al., 2019; Fujimoto & Gu, 2021). Algorithms more specific to offline setting are then further developed, motivated by the challenges residing in the offline setting (Levine et al., 2020; Kumar et al., 2020). Besides algorithmic connections, offline and online RL are also complementary to each other in terms of strengths and weaknesses. Offline RL is sample efficient since no online interactions are required, but the performance is bounded by the fixed data set. Online RL enjoys more opportunities for performance improvement, but is comparatively much less sample efficient. Because of the connections and complementary strengths, instead of treating them as two isolated topics, it is more natural to connect both in pursuit of a performant policy in practice.

The Direct Offline-Online Approach. Because of the above mentioned connections, it is tempting to directly use the same algorithm (thus the same network architectures as well) for both phases. Unfortunately, this is ineffective in *either directions*: either directly using offline RL algorithms for online learning (*forward* direction), or directly using online RL algorithms for offline learning (*reverse* direction). The reverse direction has been explored extensively in the offline RL community. The current wisdom is that instead of directly using an existing online RL algorithm (e.g. TD3 (Fujimoto et al., 2018)), special treatments need to be incorporated into the algorithm (e.g. incorporating BC into TD3 (Fujimoto & Gu, 2021)) for handling the challenges arising in offline learning, due to issues such as querying the critic function with out-of-distribution actions.

In the *forward* direction, as noted in previous work (Nair et al., 2020), it is exceptionally difficult to first train a policy using offline data and then further improve it using online RL. There are some efforts in the literature on directly transferring the parameters learned offline for online fine-tuning, *i.e.*, by initializing the policy in Eqn.(1) with parameters learned offline. This scheme is illustrated in Figure 1 as the *Direct* approach for Offline-to-Online RL. While simple, this approach has several potential issues as noted in the literature. For example, one common issue is that the behavior of the offline policy can be compromised or even destroyed at the initial phase of online training, e.g. because of the noisy gradient for policy update due to cold start learning of critic network (Uchendu et al., 2022) (as in the case of reward-free pre-training) or distribution shift between offline and

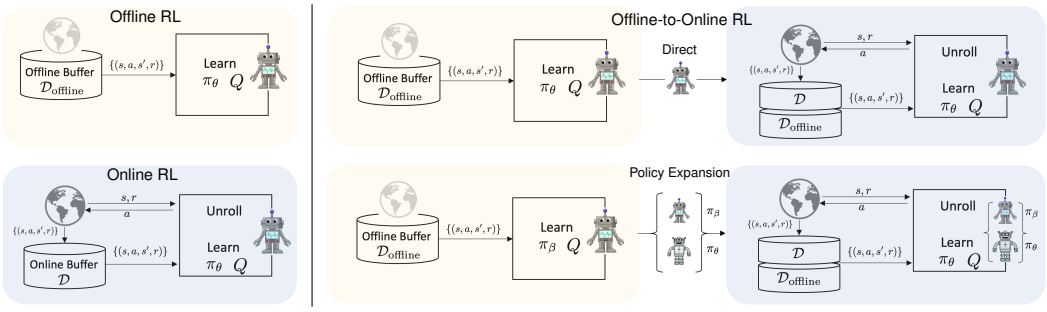


Figure 1: **Illustration of Different Training Schemes**. Offline training and online RL have been developed within their own training stages. Direct Offline-Online learning approach continues the online training stage after the offline stage is finished, updating the same policy network. The proposed Policy Expansion approach bridges offline and online training by retaining the policy after offline learning (π_β), and expand the policy set with another learnable policy (π_θ) for capturing further performance improvements. The two policies both participate in interactions with environment and learning in an adaptive fashion.

online dataset (Lee et al., 2021). Another issue is the conservativeness of the offline RL algorithms. While this is a desirable feature when considering only offline training, it is not preferred for online learning, where exploration is valuable for a further improvement (Rezaeifar et al., 2022). This is a phenomenon that is commonly observed and reported in the literature (Lee et al., 2021; Campos et al., 2021; Uchendu et al., 2022).

4 Bridging Offline and Online RL via Policy Expansion

In this section, we will introduce a simple scheme called Policy Expansion for bridging offline and online training. It is worthwhile to note that the proposed scheme is orthogonal to specific off/online RL algorithms and is generally applicable to existing offline and online algorithms.

4.1 Policy Expansion and Adaptive Composition

Policy Expansion. To mitigate the above mentioned issues, we propose an alternative scheme that can be readily combined with existing algorithms. The proposed approach is illustrated in Figure 1. Given a policy π_β obtained from offline training phase, instead of directly fine-tuning the parameters, we freeze π_β and add it into a policy set $\Pi = [\pi_\beta]$. To enable further learning, instead of directly modifying π_β as in the *Direct* method, which has the potential of destroying useful behaviors learned offline, we freeze π_β and expand the policy set Π with another learnable policy π_θ as

$$\Pi = [\pi_\beta, \pi_\theta] \tag{4}$$

which is responsible for a further performance improvement during online training. We refer this type of policy construction as Policy Expansion (PEX). It is intuitive to understand that the behavior of offline policy π_β is free from being negatively impacted, while the newly added policy can be updated. The policies in the policy set Π will all get involved into exploration and learning in an collaborative and adaptive manner as detailed in the following.

Adaptive Policy Composition. Both policies in the policy set Π will form a single composite policy $\tilde{\pi}$, which will be used in both exploration and learning. More specifically, given the current state s, we first sample actions for each member of the policy set Π and form a proposal actions set $\mathbb{A} = \{a_i \sim \pi_i(s) | \pi_i \in \Pi\}$. Then all the action proposals will be taken into consideration and they will be selected with the probability related to their potential utilities (e.g. values). For example, we can compute their values at the current state $\mathbf{Q}_\phi = [Q_\phi(s, a_i) | a_i \in \mathbb{A}] \in \mathbb{R}^K$, with K denotes the cardinality of Π (here $K = 2$), and construct a categorical distribution for selecting the final action:

$$P_{\mathbf{w}}[i] = \frac{Q_\phi(s, a_i)/\alpha}{\sum_i Q_\phi(s, a_i)/\alpha}, \quad \forall i \in [1, \cdots K] \tag{5}$$

where α is temperature. Then we can sample $\mathbf{w}$ from it for selecting the actions $\mathbf{w} \sim P_{\mathbf{w}}$ to decide which action will be used during unroll for interacting with environment. Using value for policy composition has been used in different contexts in the literature (Yu et al., 2021; Shah et al., 2022).

Conceptually, the composite policy $\tilde{\pi}$ can be represented as follows:

$$\tilde{\pi}(a|s) = [\delta_{a\sim\pi_\beta(s)}, \delta_{a\sim\pi_\theta(s)}]\mathbf{w}, \quad \mathbf{w} \sim P_{\mathbf{w}} \tag{6}$$

where $\mathbf{w} \in \mathbb{R}^K$ a one-hot vector, indicating the policy that is selected for the current state s. $\delta_{a\sim\pi}$ denotes the Dirac delta distribution centered at a which is sampled from π. By allowing only the newly added policy (π_θ in this case) to be fine-tuned while freezing all others (π_β), we can avoid the problem of compromising (e.g. destroying or forgetting) the behavior of offline policy. At the same time, we have the advantage of adaptiveness in the sense of allowing learning of new abilities. From this perspective, the policy expansion plays the role of bridging the offline and online learning phases, while mitigating commonly encountered issues. It is interesting to note that a similar compositional form of policy has appeared in DAgger (Ross et al., 2011), although with a uniform weight across states and under a different context of imitation learning. Here our compositional weight is state adaptive and the compositional policy is used for bridging offline-to-online reinforcement learning.

PEX has several advantages compared to direct offline-online learning (illustrated in Figure 1):

1. *offline policy preservation*: it can retain the useful behaviors learned during offline training phase by retaining the policy and avoid it being destroyed in the initial online training phase;
2. *flexibility in policy form*: the offline policy does not need to be of the same form with the online policy (e.g. same network structure) as in the direct offline-online approach, offering more flexibilities in design;
3. *adaptive behavior selection*: both the behavior of the offline policy and the online learning policy are used in interacting with the environment and they are involved in an adaptive manner, e.g., according to their respective expertise in handling different states.

Algorithm 1 PEX: Policy Expansion for Offline-to-Online RL

Input: offline RL algorithm $\{L^{Q_\phi}_{\text{offline}}, L^{\pi_\beta}_{\text{offline}}\}$, online RL algorithm $\{L^{Q_\phi}_{\text{online}}, L^{\pi_\theta}_{\text{online}}\}$[1]
Initialize: network parameters ϕ, β, θ, offline replay buffer $\mathcal{D}_{\text{offline}}$
while in *offline training phase* **do**
  % offline policy training using batches from the offline replay buffer $\mathcal{D}_{\text{offline}}$
  $\phi \leftarrow \phi - \lambda_Q \nabla_\phi L^{Q}_{\text{offline}}(\phi), \quad \beta \leftarrow \beta - \lambda_\pi \nabla_\beta L^{\pi_\beta}_{\text{offline}}(\beta)$
end while
Policy Expansion: $\tilde{\pi} = [\pi_\beta, \pi_\theta]$; transfer Q_ϕ
while in *online training phase* **do**
  for each environment step **do**
    $a_t \sim \tilde{\pi}(a_t|s_t)$ according to (6), $s_{t+1} \sim T(s_{t+1}|s_t, a_t)$, $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$
  end for
  for each gradient step **do**
    % online training using batches from both $\mathcal{D}_{\text{offline}}$ and $\mathcal{D}$
    $\phi \leftarrow \phi - \lambda_Q \nabla_\phi L^{Q}_{\text{online}}(\phi), \quad \theta \leftarrow \theta - \lambda_\pi \nabla_\theta L^{\pi_\theta}_{\text{online}}(\theta)$
  end for
end while

4.2 Bridged Offline-Online Training with Policy Expansion

We focus on value-based RL algorithms for both stages in this work. For *offline-training*, we conduct offline RL with an offline RL algorithm (e.g. IQL) on the offline dataset to obtain the offline policy π_β. Then we can construct policy expansion following Eqn.(4) before entering the online phase. And we also transfer the Q function (critic) learned in the offline stage to online stage for further learning. We also transfer the offline buffer to online stage as an additional buffer, as shown in Figure 1.

For *online training*, we use the policy adaptively composed from the policy set as in Eqn.(6), and then conduct online training by interleaving environmental interaction and gradient update. The newly collected transitions are stored into the online replay buffer $\mathcal{D}$. For training, batches randomly sampled from both $\mathcal{D}$ and $\mathcal{D}_{\text{offline}}$ are used. The value loss and policy loss are calculated based on the losses corresponding to the chosen algorithm. The proposed scheme can be used together with different existing RL algorithms. The complete procedure is summarized in Algorithm 1, taking an offline RL and online RL algorithm as inputs.

[1]We represent a value-based RL algorithm succinctly with a pair of value and policy losses as $\{L^Q, L^\pi\}$.

5 Related Work

Pre-Training in RL. A number of different directions have been explored in RL pre-training, including representation pre-training and policy pre-training. Note that for RL, pre-training can be either offline or online. Representative works include pre-training of the feature representation using standard representation learning methods (e.g. contrastive learning (Yang & Nachum, 2021)), dynamics learning-based representation learning (e.g. Schwarzer et al., 2021; Seo et al., 2022), unsupervised RL driven by intrinsic rewards in reward-free environments (e.g. Liu & Abbeel, 2021), or directly using ImageNet pre-training for visual RL tasks (e.g. Shah & Kumar, 2021; Yuan et al., 2022). Apart from representation pre-training, another category of work is on policy pre-training, with the goal of acquiring behaviors during the pre-training phase that are useful for the online phase. When the downstream task is unknown, there are approaches for unsupervised pre-training, e.g., maximizing behavior diversity (Eysenbach et al., 2019) or converting the action space Singh et al. (2021), with the hope of discovering some behaviors that are useful for the downstream task. When the offline-online task is more aligned, there are some early attempts on directly transferring policy parameters (Rajeswaran et al., 2018), based on the intuition that a policy initialized this way can produce more meaningful behaviors than randomly initialized networks. Some recent work focuses on behavior transferring, *i.e.*, leveraging the offline trained policy for exploration during online training (Campos et al., 2021; Uchendu et al., 2022). Our work falls into this latter category of methods. One notable difference compared to Campos et al. (2021); Uchendu et al. (2022) is that for the proposed approach, the offline policy is one part of the final policy, and with its role been determined adaptively.

Data-Driven RL and Offline RL. Training a policy by leveraging a large amount of existing data (*i.e.* data-driven RL) is a promising approach that is valuable to many real-world scenarios, where offline data is abundant. Offline RL is one active topic towards this direction. The main motivation of offline RL is to train a policy by leveraging a pre-collected dataset, without requiring additional environmental interactions (Levine et al., 2020). Many research works in this direction focus on addressing the special challenges brought by offline learning, including out-of-distribution value issue (Kumar et al., 2020). Common strategies include constraining the value (Kumar et al., 2020) to be small for out of distribution actions or the policy to be close to the action distribution of dataset (Fujimoto et al., 2019; Fujimoto & Gu, 2021). Recently, Kostrikov et al. (2022) proposes an implicit Q-learning (IQL) method as an alternative way to handle this issue. It learns a value function that predicts a certain expectile of the values for state-action pairs from the dataset, which can be used for computing the value target without querying the critic with out-of-distribution actions.

Offline Training with Online Fine-tuning. Combining offline data with online learning is an effective approach that have been demonstrated by several early attempts with demonstration data (Vecerík et al., 2017; Hester et al., 2018; Nair et al., 2018; Rajeswaran et al., 2018). Traditionally, the offline RL methods are purely focused on the offline training setting. However, the offline learned policy could be limited in performance given a fixed dataset. In the case when further online interaction is allowed, it would be natural to fine-tune the policy further with data collected online. Several different approaches have been explored towards online fine-tuning of a offline pre-trained policy, including balancing offline-online replay data (Lee et al., 2021), parameter transferring (Rajeswaran et al., 2018; Xie et al., 2021), policy regularization (Rudner et al., 2021; Tirumala et al., 2020) and guided exploration (Campos et al., 2021; Uchendu et al., 2022). It has been observed that directly applying some offline RL methods does not benefit from online interactions (Nair et al., 2020; Uchendu et al., 2022), potentially due to the conservative nature of the offline policy (Nair et al., 2020). Based on this observation, there are some recent efforts on developing algorithms that are not only suitable for offline training, but also can leverage online interactions for further learning. Nair et al. (2020) shows that an advantage-weighted form of actor-critic method is suitable for this purpose. IQL (Kostrikov et al., 2022) also leverages a similar form for policy learning and shows that it can benefit from online fine-tuning. Our work falls into this category and we provide an alternative approach for leveraging offline pre-training for helping with online training.

6 Experiments

In this section, we first evaluate the effectiveness of the proposed approach on various types of benchmark tasks with comparison to a number of baseline methods. Then we further show a number of extensions where the proposed approach can also be applied.

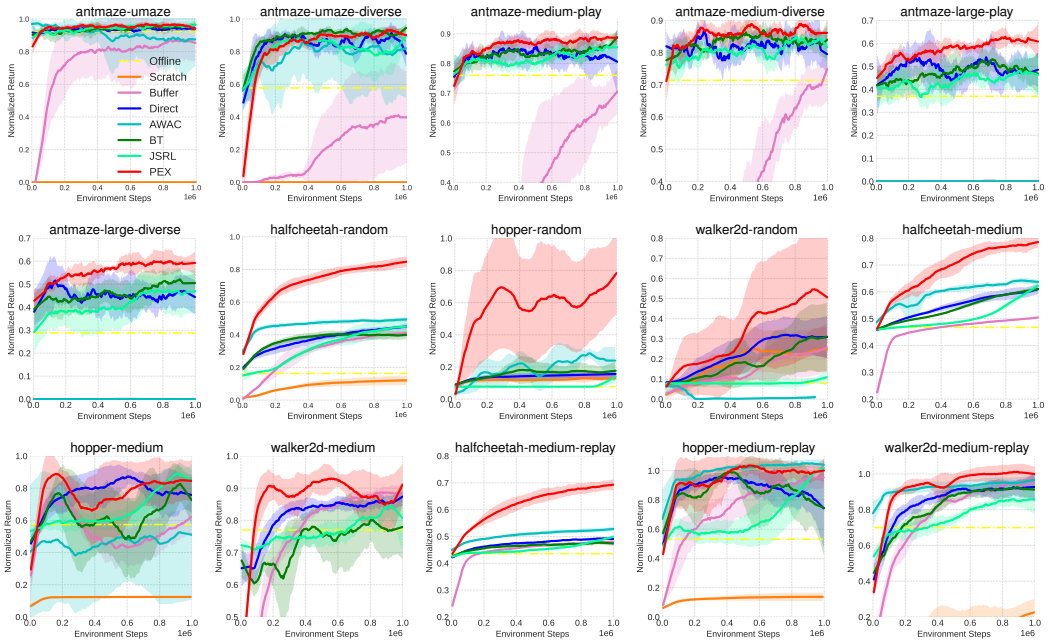


Figure 2: **Normalized Return Curves** of different methods on benchmark tasks from D4RL (Fu et al., 2020). IQL is used as the backbone apart from the AWAC baseline (Nair et al., 2020).

6.1 Offline-to-Online RL Experiments

Tasks and Settings. We use the standard D4RL benchmark which has been widely used in offline RL community (Fu et al., 2020). For offline learning, we use the provided dataset for training. For online learning, we use the accompanied simulator for interaction and training. For offline phase, 1M training steps are used for training. Then we run online fine-tuning for another 1M environmental steps. Here we use IQL (Kostrikov et al., 2022) as the backbone algorithm for all methods listed below. The training is repeated with 5 different random seeds.

Baselines. We compare the proposed approach with the following baselines: ***(i)*** **`Offline`**: offline training using IQL, without online fine-tuning; ***(ii)*** **`Scratch`**: train IQL online from scratch, without offline-pre-training. ***(iii)*** **`Buffer`** (Vecerík et al., 2017): train IQL online without offline pre-training, but has access to the offline buffer during online training, *i.e.* using a buffer as $\mathcal{D} \cup \mathcal{D}_{\text{offline}}$. ***(iv)*** **`Direct`** (Kostrikov et al., 2022): a direct offline-to-online approach by directly transferring parameters trained offline to online stage using IQL (Kostrikov et al., 2022), which is a recent and representative RL algorithm that shows state-of-the-art performance on offline RL while allows online fine-tuning; ***(v)*** **`AWAC`** (Nair et al., 2020): an approach that uses an advantage-weighted form of actor-critic method for offline-to-online RL; ***(vi)*** **`Off2On`** (Lee et al., 2021): a recent offline-to-online RL method that uses an ensemble of offline trained value and policies together with a balanced offline-online replay scheme; ***(vii)*** **`BT`** (Campos et al., 2021): Behavior Transfer which is an approach that leverages the an offline learned policy in exploration, where the offline policy is used for exploration for a consecutive number of steps sampled from a distribution once activated; ***(viii)*** **`JSRL`** (Uchendu et al., 2022): Jump Start RL, which divides the rollout of a trajectory into two parts, using the offline learned policy for the first part and then unrolling with the online learning policy for the rest of trajectory. **`PEX`** denotes the proposed approach, which has the same offline and online RL algorithms as **`Direct`**, and is only different in using *Policy Expansion* for connecting the two stages. The code will be released. [2]

The return curves for all the tasks are shown in Figure 2. The aggregated return across all tasks is shown in Figure 3. The returns are first averaged across task and then across runs. It can be observed that all methods show some improvements after online training in general, compared to the initial performance before online training. **`Scratch`**

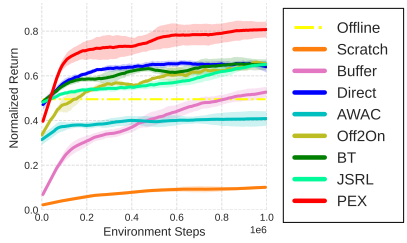


Figure 3: **Aggregated Return Curves** across tasks (IQL-based).

[2] https://github.com/Anonymous/PEX_Placeholder_URL

has the lowest overall performance across all tasks. On the challenging sparse reward antmaze tasks, **Scratch** cannot learn a meaningful policy at all. **Buffer** has better performance than **Scratch** when incorporating the offline buffer, indicating that the offline buffer has some benefits in helping with learning. Notably, on the previously zero-performance antmaze tasks, **Buffer** achieves reasonable performance with the help of the offline buffer, leading to a large overall improvement over **Scratch**, (*c.f.* Figure 3). **Direct** (IQL-based) shows large improvements over **Offline** on average as shown in Figure 3, implying the benefits brought by the the additional online training over pure offline training. **BT** shows some improvements over **IQL** on some tasks such as `antmaze-medium-play` and `antmaze-large-diverse`, with an overall performance comparable to that of **Direct**. **JSRL** outperforms **Direct** and **BT** on some tasks (e.g. `hopper-medium`, `hopper-medium-replay`), potentially due to its different way of leveraging the offline policy, and its overall performance is similar to **BT**. The proposed **PEX** approach performs comparably to other baselines on some tasks while outperforming all baseline methods on most of the other tasks (*c.f.* Figure 2), and outperforms baselines methods overall (*c.f.* Figure 3), demonstrating its effectiveness.

6.2 Heterogeneous Offline-Online RL Bridging via Policy-Expansion

We have shown the application of **PEX** to the case where both the offline and online algorithms are the same (referred to as **PEX-IQL** here since both are IQL) in Section 6.1. In this section, we further show the applicability of the proposed scheme in bridging heterogeneous RL algorithms, *i.e.* different RL methods are used for the offline and online stages. As an example, here we use IQL (Kostrikov et al., 2022) and SAC (Haarnoja et al., 2018) for offline and online RL stages respectively. We compare with **Scratch** (vanilla SAC (Haarnoja et al., 2018)), **Buffer** (SAC with additional offline replay buffer) as well as **Direct** (directly transferring policy and critic parameters learned with offline IQL to SAC for further online learning). Again **PEX** uses the same offline and online algorithms as in **Direct** but uses policy expansion instead of the direct transferring approach. The normalized return curves aggregated across tasks are shown in Figure 4. Individual return curves are shown in Appendix A.8. It can be observed that there is an overall improvement by simply applying **PEX** to the heterogeneous offline-online RL setting as well.

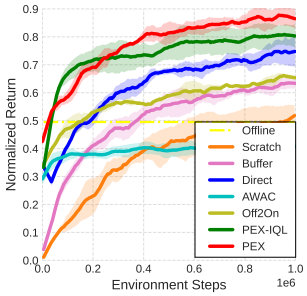


Figure 4: **Aggregated Return Curves across benchmark tasks (SAC-based)**.

6.3 Ablation Studies

We will inspect the impact of several factors on the performance of the proposed method in the sequel.

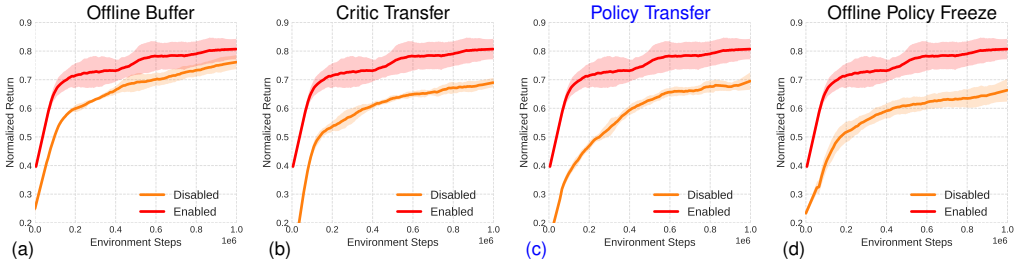


Figure 5: **Ablation Results** on a number of factors. Orange curves correspond to ablation variants.

Offline-Buffer. This experiment investigates the impacts of including the offline replay buffer in online training stage. The results are shown in Figure 5(a). As can be observed, the inclusion of the offline replay buffer helps with the performance, but the performance drop caused by disabling offline replay buffer is smaller than the change of other algorithmic components (*c.f.* Figure 5(b)∼(d)).

Critic Transfer. This experiment studies the impacts of transferring the critic parameters trained from offline to online stage. As can be observed from Figure 5(b), disabling critic transferring will also greatly decrease the performance both in terms of sample efficiency as well as final performance.

Policy Transfer. This experiment examines the impacts of transferring the offline pre-trained policy in the online training stage. When policy transferring is disabled, there is no need to use policy

expansion in the online training stage. The results are shown in Figure 5(c). It can be observed that there is a clear performance drop when policy transferring is disabled.

Offline Policy Freeze. The offline learned π_β is freezed during the online learning stage in **PEX**. This experiment investigates the impact of this factor. If disabled, π_β will be trained in the same way as π_θ during online learning. It is observed from Figure 5(d) that freezing the offline policy is important. Training by disabling policy freezing has a clear performance drop. This is consistent with the intuition on the usefulness of offline policy preservation, which is one advantage of our approach. More ablation results are deferred to Appendix A.4.

6.4 Visualization and Analysis

Policy Composition Probability. Since a composite policy is involved in the proposed approach, it is interesting to inspect the participation of each member policy from the policy set $\Pi = \{\pi_\beta, \pi_\theta\}$ when interacting with the environment. We visualize the policy compositional probability $P_{\mathbf{w}}$ during the rollout within a trajectory after training, as shown in Figure 6. It can be observed that composition probability for each member policy is state-adaptive and is changing along the progress of a trajectory, implying that both policies contribute to the final policy in an adaptive manner.

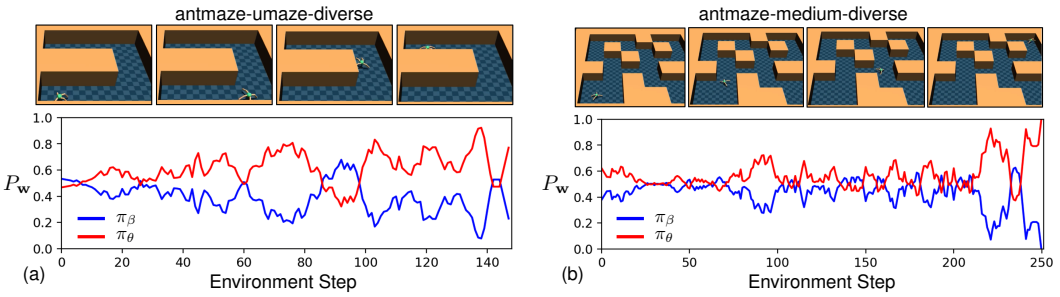
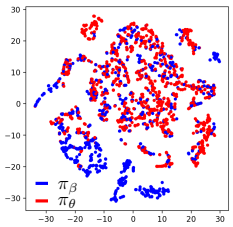


Figure 6: **Visualization of Policy Composition Probability** during the rollout of one trajectory. The probability curves are smoothed for visualization purpose.

State Space Associations of Member Policies. To get a better understand of the role of the member policies in the policy set, we visualize the association between the states and its selected policy. For this purpose, we embed a set of states into a 2D space using t-SNE (van der Maaten & Hinton, 2008), and then visualize the association of offline policy π_β and the newly expanded policy π_θ to states in the projected space. States that select the offline policy π_β are colored with blue and states that select π_θ are colored with red. It can be observed that π_θ and π_θ cover different parts of the state space, indicating that they have some complementary functionalities and are preferred differently at different states.



7 Conclusions, Limitations and Future Work

We highlight the usefulness of properly connecting offline and online stages of reinforcement learning to gain the benefits of both worlds and present a policy expansion scheme as an attempt toward this direction. This scheme is an instance in the direction orthogonal to developing completely new offline-to-online RL algorithms. The proposed approach is simple and can be combined with existing RL algorithms, and is illustrated with two different combinations in this work. Experiments demonstrate the effectiveness of the proposed approach on a number of benchmark tasks.

While achieving promising performance, the proposed approach also has some limitations. One limitation is that the number of parameters grows with the number of policies in the policy set. While this might not be an issue when the policy set is small as the case in this work, it will be less parameter efficient in the presence of a large policy set. One possible way to address this issue is by introducing a distillation stage (Rusu et al., 2016), by consolidating the multiple policies into a network with a smaller number of parameters. Currently the pre-trained policy is a single wholistic policy that operates at the primitive level of actions. Generalizing the proposed scheme to the case with a set of pre-trained skill policies is an interesting direction (Eysenbach et al., 2019; Shu et al., 2018). The idea of using a policy set itself can potentially be applied to other cases beyond offline to online RL. We leave the exploration of its application to other cases as an interesting future work.

REFERENCES

Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, 1996.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott E. Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, Mel Vecerík, Oleg Sushkov, David Barker, Jonathan Scholz, Misha Denil, Nando de Freitas, and Ziyu Wang. A framework for data-driven robotics. *CoRR*, abs/1909.12200, 2019.

Víctor Campos, Pablo Sprechmann, Steven Hansen, André Barreto, Steven Kapturowski, Alex Vitvitskyi, Adrià Puigdomènech Badia, and Charles Blundell. Beyond fine-tuning: Transferring behavior in reinforcement learning. *CoRR*, abs/2102.13515, 2021.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, 2021.

Felipe Codevilla, Eder Santana, Antonio M. López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *International Conference on Computer Vision*, 2019.

Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. In *Advances in Neural Information Processing Systems*, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: datasets for deep data-driven reinforcement learning. *CoRR*, abs/2004.07219, 2020.

Scott Fujimoto and Shixiang Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 2018.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, 2019.

Weifeng Ge and Yizhou Yu. Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Dibya Ghosh, Anurag Ajay, Pulkit Agrawal, and Sergey Levine. Offline RL policies should be trained to be adaptive. In *International Conference on Machine Learning*, 2022.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, 2017.

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*, 2018.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, Gabriel Dulac-Arnold, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Deep Q-learning from demonstrations. In *AAAI Conference on Artificial Intelligence*, 2018.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*, 2021.

Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better ImageNet models transfer better? In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. Should I run offline reinforcement learning or behavioral cloning? In *International Conference on Learning Representations*, 2022.

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Annual Conference on Robot Learning*, 2021.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020.

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. In *International Conference on Learning Representations*, 2016.

Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. In *Advances in Neural Information Processing Systems*, 2021.

Cong Lu, Philip J. Ball, Tim G. J. Rudner, Jack Parker-Holder, Michael A. Osborne, and Yee Whye Teh. Challenges and opportunities in offline reinforcement learning from visual observations. *CoRR*, abs/2206.04779, 2022.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

Urs Muller, Jan Ben, Eric Cosatto, Beat Flepp, and Yann LeCun. Off-road obstacle avoidance through end-to-end learning. In *Advances in Neural Information Processing Systems*, 2005.

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

Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. AWAC: Accelerating online reinforcement learning with offline datasets. *CoRR*, abs/2006.09359, 2020.

Dean A. Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*, 1988.

Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. In *Tech. Report*, 2018.

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 *Proceedings of Robotics: Science and Systems*, 2018.

Hossein Rastgoftar, Bingxin Zhang, and Ella M. Atkins. A data-driven approach for autonomous motion planning and control in off-road driving scenarios. In *Annual American Control Conference*, 2018.

Shideh Rezaeifar, Robert Dadashi, Nino Vieillard, Léonard Hussenot, Olivier Bachem, Olivier Pietquin, and Matthieu Geist. Offline reinforcement learning as anti-exploration. In *AAAI Conference on Artificial Intelligence*, 2022.

Stephane 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*, 2011.

Tim G. J. Rudner, Cong Lu, Michael Osborne, Yarin Gal, and Yee Whye Teh. On pathologies in KL-regularized reinforcement learning from expert demonstrations. In *Advances in Neural Information Processing Systems*, 2021.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In *International Conference on Learning Representations*, 2016.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. Mastering Atari, Go, Chess and Shogi by planning with a learned model. *CoRR*, abs/1911.08265, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron C Courville. Pretraining representations for data-efficient reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

Younggyo Seo, Kimin Lee, Stephen L James, and Pieter Abbeel. Reinforcement learning with action-free pre-training from videos. In *International Conference on Machine Learning*, 2022.

Dhruv Shah, Alexander T Toshev, Sergey Levine, and Brian Ichter. Value function spaces: Skill-centric state abstractions for long-horizon reasoning. In *International Conference on Learning Representations*, 2022.

Rutav Shah and Vikash Kumar. RRL: Resnet as representation for reinforcement learning. In *International Conference on Machine Learning*, 2021.

Tianmin Shu, Caiming Xiong, and Richard Socher. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. In *International Conference on Learning Representations*, 2018.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. In *International Conference on Learning Representations*, 2021.

R. S. Sutton, D. Mcallester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 2000.

Dhruva Tirumala, Alexandre Galashov, Hyeonwoo Noh, Leonard Hasenclever, Razvan Pascanu, Jonathan Schwarz, Guillaume Desjardins, Wojciech Marian Czarnecki, Arun Ahuja, Yee Whye Teh, and Nicolas Heess. Behavior priors for efficient reinforcement learning. *CoRR*, 2020.

Pedro A. Tsividis, João Loula, Jake Burga, Nathan Foss, Andres Campero, Thomas Pouncy, Samuel J. Gershman, and Joshua B. Tenenbaum. Human-level reinforcement learning through theory-based modeling, exploration, and planning. *CoRR*, abs/2107.12544, 2021.

Ikechukwu Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matthew Bennice, Chuyuan Fu, Cong Ma, Jiantao Jiao, Sergey Levine, and Karol Hausman. Jump-start reinforcement learning. In *arXiv*, 2022.

Laurens van der Maaten and Geoffrey E. Hinton. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

Matej Vecerík, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin A. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *CoRR*, abs/1707.08817, 2017.

Chuan Wen, Jierui Lin, Trevor Darrell, Dinesh Jayaraman, and Yang Gao. Fighting copycat agents in behavioral cloning from observation histories. *CoRR*, abs/2010.14876, 2020.

Tengyang Xie, Nan Jiang, Huan Wang, Caiming Xiong, and Yu Bai. Policy finetuning: Bridging sample-efficient offline and online reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

Haoran Xu, Xianyuan Zhan, Honglei Yin, and Huiling Qin. Discriminator-weighted offline imitation learning from suboptimal demonstrations. In *International Conference on Machine Learning*, 2022.

Chao-Han Huck Yang, Zhengling Qi, Yifan Cui, and Pin-Yu Chen. Pessimistic model selection for offline deep reinforcement learning. *CoRR*, abs/2111.14346, 2021.

Mengjiao Yang and Ofir Nachum. Representation matters: Offline pretraining for sequential decision making. In *International Conference on Machine Learning*, 2021.

Haonan Yu, Wei Xu, and Haichao Zhang. TAAC: Temporally abstract actor-critic for continuous control. In *Advances in Neural Information Processing Systems*, 2021.

Zhecheng Yuan, Zhengrong Xue, Bo Yuan, Xueqian Wang, Yi Wu, Yang Gao, and Huazhe Xu. Pre-trained image encoder for generalizable visual reinforcement learning. In *First Workshop on Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML*, 2022.

Qinqing Zheng, Mikael Henaff, Brandon Amos, and Aditya Grover. Semi-supervised offline reinforcement learning with action-free trajectories. *CoRR*, abs/2210.06518, 2022.

A Appendix

A.1 Details on Baselines

In this section, we provide more details on the baseline methods.

Behavior Transfer (`BT`). This approach is originally proposed for the case of pre-training using unsupervised reinforcement learning and then transfers the learned behaviors for later learning. It is also originally used in discrete action cases. Here we adapt this idea to build a baseline in our setting based on the approach of transferring behaviors in (Campos et al., 2021). Following (Campos et al., 2021), a Zeta distribution with parameter $a = 2$ is used for determining the number of steps of persistent unroll using the offline policy. The number of persistent unroll steps is re-sampled when currently not in the middle of a persistent unroll and with a random number sampled from $[0, 1]$ is smaller than a threshold ϵ. $\epsilon = 0.1$ is used in experiments.

Jump Start RL (`JSRL`). `JSRL` divides the complete unroll of one trajectory into two parts, with the first part obtained with the offline policy π_β, while the second part is obtained with the current learning policy π_θ. The core idea of `JSRL` is to initially make the first part longer, and then progressively reducing the length of the first part, thus forming a type of curriculum to help with exploration and learning. We follow this core idea by using a linear scheduler that anneals from the max episode length to 0 for progressively decreasing the first stage of unroll using the prior policy π_β.

Offline-to-Online RL (`Off2On`). `Off2On` is an approach that aims to improve offline to online RL performance (Lee et al., 2021). It incorporates two components to cope with the challenges of offline-to-online RL, such as the catastrophic performance drop: 1) an ensemble of critic networks and policy networks are trained during offline training stage, in contrast to typical offline RL, which only learns one policy network and critic network [3]. Following the setting in Lee et al. (2021), ensemble size of 5 is used. 2) balanced replay: to mitigate the negative effects of We used the code released by the authors of (Lee et al., 2021). [4]

A.2 Implementation Details on Policy Expansion

Policy Expansion (`PEX` with IQL-backbone). The proposed scheme can be combined with many different existing RL methods. Here we illustrate with an concrete instantiation using IQL (Kostrikov et al., 2022). For pre-training, we conduct offline RL with IQL on the offline dataset to obtain the offline policy π_β. Then we conduct policy expansion as Eqn.(6) before switching to online phase.

For *offline training*, we follow IQL (Kostrikov et al., 2022) to train the parameters of the value function using the TD-learning approach as follows:

$$L_Q(\phi) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}_{\text{offline}}} \|Q_\phi(s,a) - (r + \gamma V(s'))\|^2 \tag{7}$$

$$L_V(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{offline}}} L_2^\tau[Q_{\bar{\phi}}(s,a) - V_\psi(s)] \tag{8}$$

where $L_2^\tau(u) = |\tau - \mathbb{1}(u < 0)|u^2$ (Kostrikov et al., 2022), with τ denotes the expectile value (Kostrikov et al., 2022). $\bar{\phi}$ denotes a set of target parameters that are periodically copied from ϕ. The policy network π_θ can be further updated by leveraging the updated critic network. Here we again follow IQL (Kostrikov et al., 2022) and use the weighted form for actor training:

$$L_\pi(\beta) = \mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{offline}}} - \lfloor w(s,a) \rfloor \cdot \log \pi_\beta(a|s), \tag{9}$$

with $w(s,a) = \exp((Q(s,a) - V(s))/\alpha)$ and $\lfloor \cdot \rfloor$ denotes the gradient stopping operator.

For *online training*, we first expand the policy set as in Eqn.(4), and then conduct online training by interleaving environmental interaction and gradient update. The newly collected transitions are saved into replay buffer $\mathcal{D}$. For training, batches randomly sampled from both $\mathcal{D}$ and $\mathcal{D}_{\text{offline}}$ are used. The values losses are the same as the offline phase, and the policy loss is computed against π_θ:

$$L_\pi(\theta) = \mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{offline}}\cup\mathcal{D}} - \lfloor w(s,a) \rfloor \cdot \log \pi_\theta(a|s), \tag{10}$$

For $a_0 \sim \pi_\beta(s_t)$, in practice, we simply take the greedy action from the policy. Following Kostrikov et al. (2022), we model the policy using a non-squashed Gaussian distribution with mean squashed to action range, and use state independent standard deviations.

Policy Expansion (`PEX` with with SAC as online backbone). The main procedure of online training is the same as SAC (Haarnoja et al., 2018), with an adaptation for the actor training by changing the part of the loss based on Q-function from $\mathbb{E}_{s\sim\mathcal{D}}\mathbb{E}_{a\sim\pi_\theta} \left\| \lfloor \frac{\partial Q(s,a)}{\partial a} + a \rfloor - a \right\|^2$ (which is an equivalent form of loss to $\mathbb{E}_{s\sim\mathcal{D}}\mathbb{E}_{a\sim\pi_\theta} - Q(s,a)$) to $\mathbb{E}_{s\sim\mathcal{D}}\mathbb{E}_{a\sim\tilde{\pi},a_0\sim\pi_\theta} \left\| \lfloor \frac{\partial Q(s,a)}{\partial a} + a \rfloor - a_0 \right\|^2$.

[3] Here we refer the commonly used double critic replica to reduce value overestimation as in SAC (Haarnoja et al., 2018) and TD3 (Fujimoto et al., 2018) as a single critic network. The ensemble used in Lee et al. (2021) is applied on top of this.

[4] `https://github.com/shlee94/Off2OnRL`

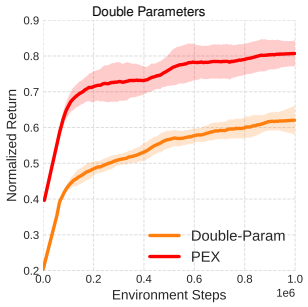
A.3 INTUITIVE INTERPRETATION OF PEX

$P_{\mathbf{w}}$ can be viewed as a discrete policy with action dimension of two, which selects between two candidate actions, one from offline policy π_β and the other from π_θ. Ideally, it assigns higher probability to actions with higher values at the current state. Eqn.(6) is essentially a reflection of this intuition, and is inspired by the energy-based policies (Haarnoja et al., 2017). During exploration, the action selection is sampled from $P_{\mathbf{w}}$. Since $Q_{\mathbf{w}}$ captures some crude knowledge of action values, the $P_{\mathbf{w}}$ constructed from it plays the role of a form of informed exploration. A better Q_ϕ will be learned via TD-learning based on the collected transitions, which in turn improves $P_{\mathbf{w}}$ since it is constructed from Q_ϕ. The level of accuracy of Q_ϕ can be roughly interpreted as the amount of prior knowledge encoded. When Q_ϕ is random, $P_{\mathbf{w}}$ will also be random, and we reduce to the tabula rasa case. In this case, learning can still happen, just no injected prior knowledge; When Q_ϕ is perfectly accurate, it has encoded all the useful knowledge already. In this case, no further learning is required (we agree with you on this). When Q_ϕ is not perfectly accurate which is the case most of the time in practice, it can still provide useful guidance on action selection in some cases, and might be less informative in some other cases. Along the learning process, Q_ϕ will be more and more accurate, and the guidance will be more and more informative. In summary, Q_ϕ encodes some prior knowledge which is useful for action section,which could help in exploration together with π_β.

A.4 ADDITIONAL ABLATIONS RESULTS

We provide more ablations experiments and results in this section.

Double Parameters. Since there are two member policies in the proposed approach. A natural question to ask is whether properly double the number of parameters of a policy could achieve similar performance. To verify this, we construct a type of policy network with (roughly) doubled number of parameters as $\pi_\theta(s) = g_{\theta_g} \circ f_{\theta_f}(s)$, where $f_{\theta_f}(s)$ encodes the observation s into a feature vector and g maps the feature vector to the action distribution. To (approximately) double the number of parameters, we implement f_{θ_f} as $f_{\theta_f}(s) \triangleq f_{\theta_1}(s) + f_{\theta_2}(s)$. f_{θ_1} is initialized by transferring parameters θ learned in the pre-train stage. The aggregated return curves are shown on the right, for doubling the number of policy parameters as described above (**`Double-Param`**), and the proposed approach. It can be observed that by comparison that simply increasing the number of parameters cannot help much with the performance, implying the importance of the proposed structure in **`PEX`**.



A.5 MORE DETAILS ON TASKS

We summarized the tasks used for experiments and their corresponding D4RL environment names in Table 1.

Task	D4RL Environment Name
`antmaze-umaze`	`antmaze-umaze-v0`
`antmaze-umaze-diverse`	`antmaze-umaze-diverse-v0`
`antmaze-medium-play`	`antmaze-medium-play-v0`
`antmaze-medium-diverse`	`antmaze-medium-diverse-v0`
`antmaze-large-play`	`antmaze-large-play-v0`
`antmaze-large-diverse`	`antmaze-large-diverse-v0`
`halfcheetah-random`	`halfcheetah-random-v2`
`hopper-random`	`hopper-random-v2`
`walker-random`	`walker-random-v2`
`halfcheetah-medium`	`halfcheetah-medium-v2`
`hopper-medium`	`hopper-medium-v2`
`walker-medium`	`walker-medium-v2`
`halfcheetah-medium-replay`	`halfcheetah-medium-replay-v2`
`hopper-medium-replay`	`hopper-medium-replay-v2`
`walker-medium-replay`	`walker-medium-replay-v2`

Table 1: **Tasks and the corresponding environment names in D4RL**.

A.6 HYPER-PARAMETERS

The hyper-parameters and values are summarized in Table 2. There are two algorithmic related hyper-parameters inherited from IQL since we instantiate PEX based on it: temperature α (corresponding to β^{-1} in IQL paper (Kostrikov et al., 2022)) and expectile value τ. The rest of the hyper-parameters are common ones in RL algorithms, including for example batch size, network structure and size, target update etc. We use the hyper-parameter values from the IQL paper (Kostrikov et al., 2022) in our experiments.

Hyper-parameters	Values
number of parallel env	1
discount	0.99
replay buffer size	1e6
batch size	256
MLP hidden layer size	[256, 256]
learning rate	3e-4
initial collection steps	5000
target update speed	5e-3
expectile value τ	0.9 (0.7)
inverse temperature α^{-1}	10 (3)
number of offline iterations	1M
number of online iterations	1M
number of iteration per rollout step	1
target entropy (SAC)	$-d$

Table 2: Hyper-parameter values. Values in brackets are used for locomotion tasks. The rest of the hyper-parameter values are shared across all tasks. d denotes the action dimension.

A.7 HYPER-PARAMETER ABLATIONS

Impact of α. α scales the Q values before constructing $P_{\mathbf{w}}$:

$$P_{\mathbf{w}}[i] = \frac{Q(s, a_i)/\alpha}{\sum_i Q(s, a_i)/\alpha} = \frac{\alpha^{-1} Q(s, a_i)}{\sum_i \alpha^{-1} Q(s, a_i)}, \quad \forall i \in [1, \cdots K]. \tag{11}$$

We show the impacts of its value on performance on two ant-maze tasks in Figure 7. We have use a fixed value ($\alpha^{-1} = 10$ for all antmaze tasks set according to the IQL paper (Kostrikov et al., 2022) (the β parameter). Here we show the impacts of different values for α^{-1} by setting the inverse temperature as $0.5\alpha^{-1}$, $1\alpha^{-1}$, $2\alpha^{-1}$. We note that other values are possible for different environments. While it is impractical to tune this parameter for each task, it is possible to make it more adaptive by auto-tuning in a way similar to the temperature turning in SAC (Haarnoja et al., 2018).

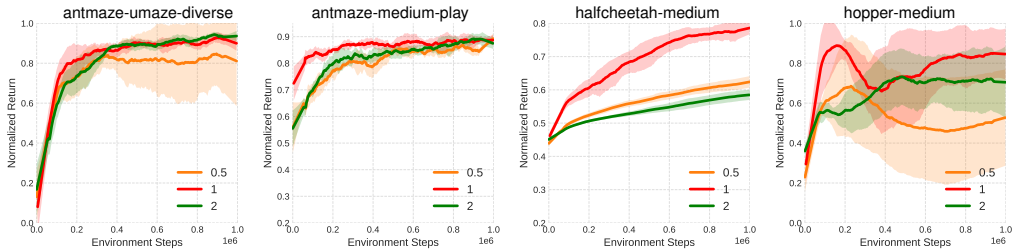


Figure 7: Impact of inverse temperature α^{-1}. Results with inverse temperature as $0.5\alpha^{-1}$, $1\alpha^{-1}$, $2\alpha^{-1}$ respectively,

Impact of Policy Entropy. The entropy of π_θ is auto-tuned to match a target entropy (Haarnoja et al., 2018), which is empirically set as $e_{\text{target}} \triangleq -d$ where d denotes the dimensionality of action. This is inherited from SAC and the same setting ($-d$) is applied to all tasks. To show the impact of different target entropy values, we experiment with three different settings: $0.5e_{\text{target}}$, $1 \cdot e_{\text{target}}$, $2e_{\text{target}}$ on several tasks. The results are summarized in Figure 8. It can be observed that overly small or large target entropy values could decrease the performance on some tasks.

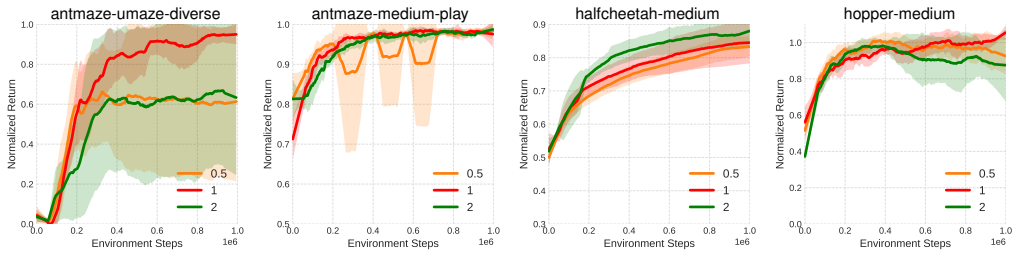


Figure 8: Impact of policy entropy (SAC-based online stage). Results with target entropy as $0.5e_{\text{target}}$, $1e_{\text{target}}$, $2e_{\text{target}}$ respectively.

A.8 More Results on Heterogeneous Offline-Online RL

Here we show detailed return curves for each task when using SAC (Haarnoja et al., 2018) as the backbone online RL algorithm. The results are shown in Figure 9.

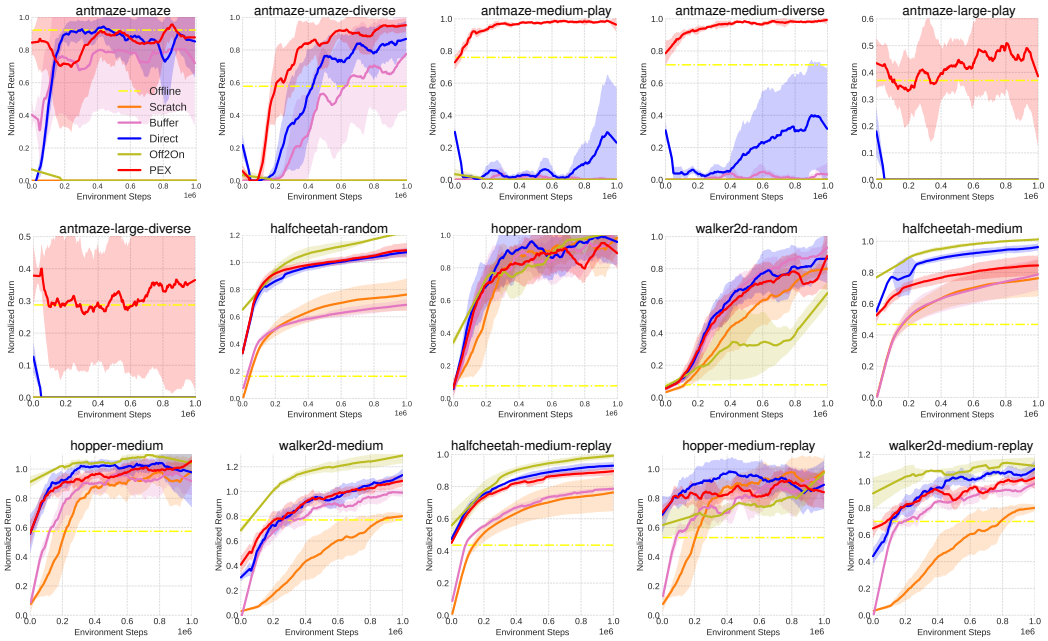


Figure 9: **Return Curves** of different methods using SAC (Haarnoja et al., 2018) as the online RL backbone algorithm, including the Off2On method (Lee et al., 2021). The returns are normalized according to D4RL benchmark (Fu et al., 2020).

A.9 Offline-Online Training with Reward-Free Offline Setting

We have shown using policy expansion for bridging offline IQL with online IQL/SAC in the main paper. Here we further demonstrate the applicability of the proposed approach by using BC as the offline learning module. We use IQL as the online module for illustration. In this section, we further demonstrate the applicability of the proposed approach to the setting with offline dataset without reward annotations. This is sometimes an encountered setting in practice, where plenty of offline data exist (e.g. from previous iteration of experiments or other agents), but without reward annotations. Since the offline dataset is only broadly related to the task and could potentially contain irrelevant or noisy data (*w.r.t.* the current task in consideration), direct behavior cloning could be sub-optimal.

Standard offline RL methods are inapplicable for offline pre-training in this case. We therefore use behavior cloning (**BC**) (Pomerleau, 1988), which is a simple and widely used approach for imitation learning. After offline pre-training using **BC**, the offline policy π_β is obtained and transferred to the

second stage of online training. In terms of Algorithm 1, only the policy loss L_{π_β} is present in the offline training phase. The offline datasets not used in online training since it has no rewards.

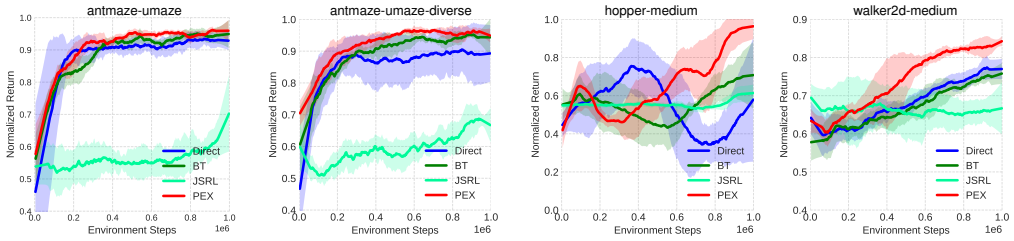


Figure 10: **Return Curves** for reward-free offline pre-training setting.

We use subset of tasks from D4RL (Fu et al., 2020) where **BC** has non-trivial performances. The results are shown in Figure 10. It can be observed that **JSRL**, which performs well under the case of having offline rewards, performs less well in the reward-free case. One potential reason is that it hinges on the offline policy to induce the state distribution (Uchendu et al., 2022). In the case where the offline policy is not highly performant, as the case here obtained from **BC**, its effectiveness could be compromised. **BT** performs comparable to or better than **Direct** online fine-tuning with IQL, potentially due to the fact that **BT** leverage the π_β in a way that has a shorter commitment length for each time the offline policy is selected. The proposed approach also shows similar online improvements comparable or sometimes better than the best performing methods.

A.10 Policy Usage

We show the policy usage of π_β and π_θ during online unroll along the process of one training session for **BT** and **JSRL** as well as **BC-PEX** (pre-trained with BC) and **IQL-PEX** (pre-trained with IQL) in Figure 11. The return curves of **PEX-BC** are shown in Figure 10. The corresponding return curves of **PEX-IQL** are shown in Figure 2. As can be observed, the usage of offline policy is changing along the course of training, and the dynamics of the usage evolution is different for different tasks.

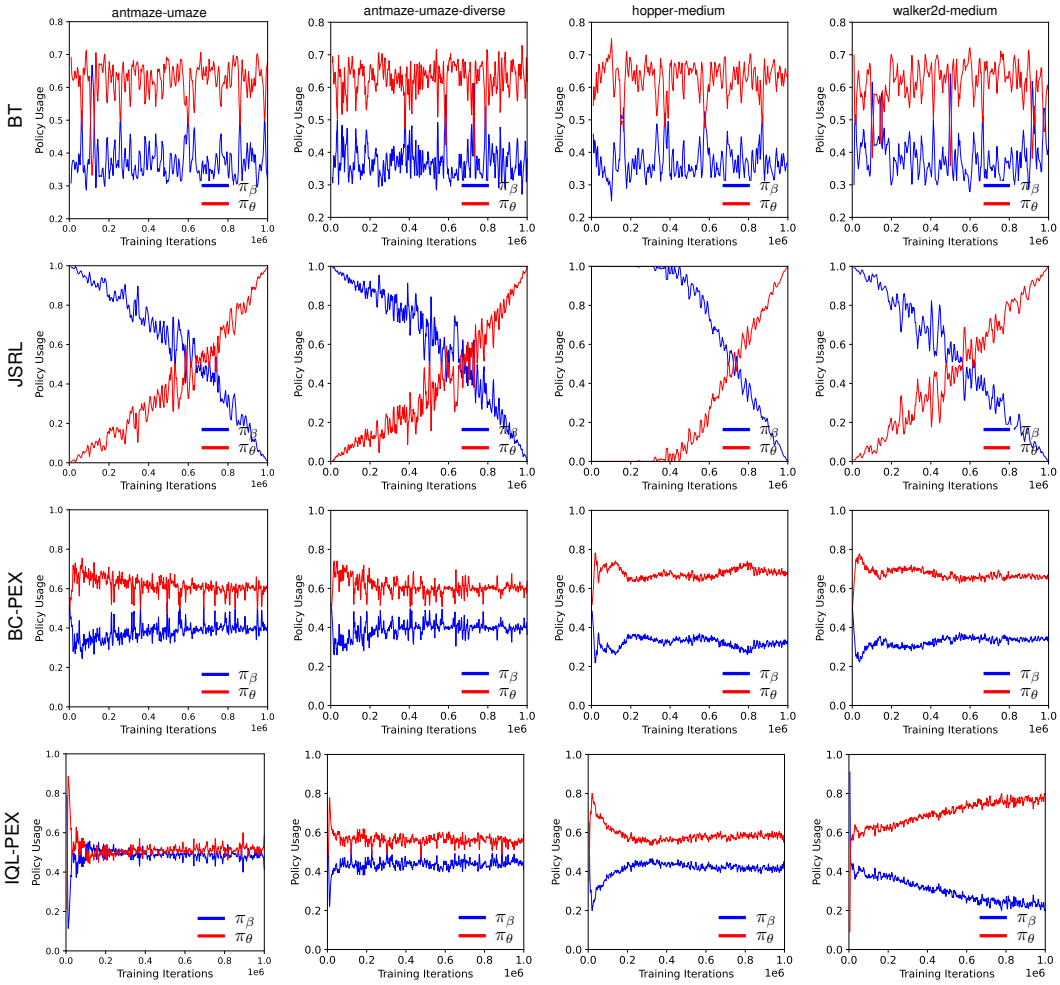


Figure 11: **Policy Usage** of π_β and π_θ along the process ofonline training for different methods.