Multi-Task Offline Reinforcement Learning with Conservative Data Sharing

Anonymous Author(s)
Affiliation
Address
email

Abstract

Offline reinforcement learning (RL) algorithms have shown promising results in domains where abundant pre-collected data is available. However, prior methods focus on solving individual problems from scratch with an offline dataset without considering how an offline RL agent can acquire multiple skills. We argue that a natural use case of offline RL is in settings where we can pool large amounts of data collected in a number of different scenarios for solving various tasks, and utilize all this data to learn strategies for all the tasks more effectively rather than training each one in isolation. To this end, we study the offline multi-task RL problem, with the goal of devising data-sharing strategies for effectively learning behaviors across all of the tasks. While it is possible to share all data across all tasks, we find that this simple strategy can actually exacerbate the distributional shift between the learned policy and the dataset, which in turn can lead to very poor performance. To address this challenge, we develop a simple technique for data-sharing in multi-task offline RL that routes data based on the improvement over the task-specific data. We call this approach conservative data sharing (CDS), and it can be applied with any single-task offline RL method. On a range of challenging multi-task locomotion, navigation, and image-based robotic manipulation problems, CDS achieves the best or comparable performance compared to prior offline multi-task RL methods and previously proposed online multi-task data sharing approaches.

1 Introduction

Recent advances in offline reinforcement learning (RL) make it possible to train policies for real-world scenarios, such as robotics [31, 57, 32] and healthcare [23, 62], entirely from previously collected data. Many realistic settings where we might want to apply offline RL are inherently *multi-task* problems, where we want to solve multiple tasks using all of the data available. For example, if our goal is to enable robots to acquire a range of different behaviors, it is more practical to collect a modest amount of data for each desired behavior, resulting in a large but heterogeneous dataset, rather than requiring a large dataset for every individual skill. Indeed, many existing datasets in robotics [16, 10] and offline RL [18] include data collected in precisely this way. Unfortunately, leveraging such heterogeneous datasets leaves us with two unenviable choices. We could train each task only on data collected for that task, but such small datasets may be inadequate for good performance. Alternatively, we could combine all of the data together, but this naïve data sharing approach can actually often degrade performance over simple single-task training [32], resulting in a brittle method with unpredictable results in practice. In this paper, we aim to understand how data sharing affects RL performance specifically in the offline setting, and develop an effective method for selectively sharing data across tasks.

A number of prior works have studied multi-task RL in the *online* setting, confirming that multi-tasking can often lead to performance that is worse than training tasks individually [53, 59, 84].

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

However, prior works have focused on mitigating optimization challenges that are aggravated by the online data generation process [61, 83, 82]. As we will find in Section 4, multi-task RL remains a challenging problem in the offline setting, particularly when sharing data across tasks, even when exploration is not an issue. While prior works have developed heuristic methods for reweighting and relabeling data [2, 15, 42, 32], they do not yet provide a principled explanation for why data sharing can hurt, especially in the offline setting, and do not provide a robust and general approach to automate selective data sharing that alleviates these issues, while still preserving the efficiency benefits of sharing experience across tasks.

In this paper, we hypothesize that data sharing can be harmful or brittle in the offline setting because it can exacerbate the distribution shift between the policy represented in the data and the policy being learned. We analyze the effect of data sharing in the offline multi-task RL setting, and present evidence to support this hypothesis. We then propose a principled approach for selective data sharing that aims to minimize distribution shift, by sharing only data that is particularly relevant to each task. Instantiating a method based on this principle requires some care, since we do not know a priori which data is most relevant for a given task before we've learned the optimal policy for that task. To provide a practical approximate instantiation of this principle, our method shares data between on estimated Q-values, relabeling transitions into a given target task when the Q-value of the added transitions exceeds the expected Q-values on the target task data. Meanwhile, we penalize the relabeled Q-values to prevent overestimation on out-of-distribution actions. The resulting algorithm, which we refer to as conservative data sharing (CDS), is able to prevent performance degradation due to excessive data sharing, while still providing large gains over naïve single-task training. We visualize how CDS works in Figure 1.

The main contribution of this work is a new algorithm, *conservative data sharing* (CDS), for multi-task offline RL problems. CDS relabels a transition into a given task only when a performance improvement over the behavior policy is expected. After data sharing, similarly to prior offline RL methods, CDS learns a conservative critic function by penalizing the value function in state-action pairs that do not lie in the support of the offline multi-task data set [37]. Further, we theoretically analyze CDS and characterize scenarios under which it provides safe policy improvement guarantees in Appendix A. Finally, we conduct extensive empirical analysis of CDS on three multi-task locomotion domains, one multi-task robotic manipulation domain with sparse rewards, one multi-task navigation domain, and one multi-task imaged-based robotic manipulation domain. We compare our method to vanilla offline multi-task RL without sharing data as well as naïvely sharing data for all tasks and existing data relabeling schemes for multi-task RL. CDS is the only method to attain good performance across all of these benchmarks, often significantly outperforming the best *domain-specific* baseline, improving over the next best baseline on each domain by 17.5% on average.

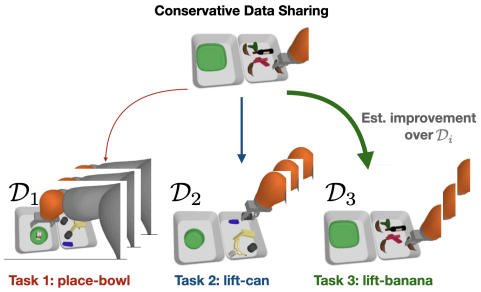

Figure 1: A visualization of CDS which shares a transition to each offline dataset $\mathcal{D}_i$ of different tasks i with a weight based on how large the estimated improvement over the behavior policy $\pi_\beta(\mathbf{a}|\mathbf{s}, i)$ of $\mathcal{D}_i$ is after sharing given transition.

2 Related Work

Offline RL [13, 58, 38, 41] refers to the problem of policy learning with a fixed dataset without collecting additional interactions in the environment. It has shown successful applications in domains such as robotic manipulation [31, 49, 57, 65, 32], NLP [28, 29], recommender systems & advertising [67, 21, 6, 73, 74] and healthcare [62, 75]. The major challenge in offline RL is the distribution shift problem [19, 35, 37], where the learned policy might generate out-of-distribution actions, resulting in erroneous value backups. Prior offline RL methods address this issue by regularizing the learned policy to be "close" to the behavior policy with the penalty on the distance between learned policy and behavior policy [19, 47, 28, 77, 87, 35, 63, 54], variants of importance sampling based algorithms [56, 69, 46, 70, 51], uncertainty quantification on Q-values [1, 35, 77, 41], learning conservative Q-functions [37, 34], and performing model-based training with penalty on out-of-distribution states [33, 85, 50, 3, 71, 57, 40, 86]. While current benchmarks in offline RL [18, 24] contains tasks that involve multi-task structure in the offline dataset, existing offline RL methods do not leverage the shared structure of multiple tasks and instead train each individual task from scratch. In this paper,

we exploit the shared structure in the offline multi-task setting and train a general policy that can successfully acquire multiple skills.

Multi-task reinforcement learning algorithms [76, 53, 72, 14, 26, 83, 80, 82, 32, 66] focus on solving multiple tasks jointly in an efficient way. While multi-task RL methods seem to provide a promising way to build general-purpose agents [32], prior works have observed major challenges in multi-task RL, in particular, the optimization challenge [26, 61, 83]. Beyond the optimization challenge, how to perform effective representation learning via weight sharing is another major challenge in multi-task RL. Prior works have considered distilling per-task policies into a single policy that solves all tasks [59, 72, 22, 80], separate shared and task-specific modules with theoretical guarantees [12] and incorporating additional supervision [66]. Finally, sharing data across tasks emerges as a challenge in multi-task RL, especially in the off-policy setting, as naïvely sharing data across all tasks turns out to hurt performance in certain scenarios [32]. In this paper, we focus on the offline setting where the challenges in data sharing are more relevant. Methods that study optimization and representation learning issues are complementary and can always be combined with our approach. We will survey methods in data sharing in multi-task off-policy RL next.

Prior works [2, 30, 55, 60, 15, 42, 32, 7] have found it effective to reuse data across tasks by recomputing the rewards of data collected for one task and using such relabeled data to impersonate other tasks, which effectively augments the amount of data available for learning each task and boost performance. These methods perform such task impersonation either uniformly [32] or based on metrics such as high Q-values of the relabeled data [15, 42], human domain knowledge [32], and distance to states or images in goal-conditioned settings [2, 55, 52, 45, 68, 44, 27, 48, 81, 7]. However, all of these methods either still require online data collection and do not consider data sharing in a fully offline setting or only solve offline goal-conditioned problems [7]. Our work extends the data sharing idea to multi-task offline RL settings and leverages conservatism used in offline RL methods [37] to ensure that data sharing improves multi-task learning performance and does not introduce excessive distribution shift.

3 Preliminaries and Problem Statement

Multi-task offline RL. The goal in multi-task RL is to find a policy that maximizes expected return in a multi-task Markov decision process (MDP), defined as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \gamma, \{R_i, i\}_{i=1}^N)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, dynamics $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$, a discount factor $\gamma \in [0, 1)$, and a finite set of task indices $1, \cdots, N$ with corresponding reward functions $R_1, \cdots, R_N$. Each task i presents a different reward function R_i, but we assume that the dynamics P are shared across tasks. While this setting is not fully general, there are a wide variety of practical problem settings for which only the reward changes including various goal navigation tasks [18], distinct object manipulation objectives [78], and different user preferences [9]. In this work, we focus on learning a policy $\pi(\mathbf{a}|\mathbf{s}, i)$, which in practice could be modelled as independent policies $\{\pi_1(\mathbf{a}|\mathbf{s}), \cdots, \pi_N(\mathbf{a}|\mathbf{s})\}$ that do not share any parameters, or as a single task-conditioned policy, $\pi(\mathbf{a}|\mathbf{s}, i)$ with parameter sharing. Our goal in this paper is to analyze and devise methods for data sharing and the choice of parameter sharing is orthogonal, and can be made independently. We formulate the policy optimization problem as finding a policy that maximizes expected return over all the tasks: $\pi^*(\mathbf{a}|\mathbf{s}, \cdot) := \arg\max_\pi \mathbb{E}_{i\sim[N]}\mathbb{E}_{\pi(\cdot|\cdot,i)}[\sum_t \gamma^t R_i(\mathbf{s}_t, \mathbf{a}_t)]$. The Q-function, $Q^\pi(\mathbf{s}, \mathbf{a}, i)$, of a policy $\pi(\cdot|\cdot, i)$ is the expected long-term discounted reward obtained in task i by executing action $\mathbf{a}$ at state $\mathbf{s}$ and following policy π thereafter.

Standard offline RL is concerned with learning policies $\pi(\mathbf{a}|\mathbf{s})$ using only a given static dataset of transitions $\mathcal{D} = \{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j)\}_{j=1}^N$, collected by a behavior policy $\pi_\beta(\mathbf{a}|\mathbf{s})$, without any additional environment interaction. In the multi-task offline RL setting, the dataset $\mathcal{D}$ is partitioned into per-task subsets, $\mathcal{D} = \cup_{i=1}^N \mathcal{D}_i$, where $\mathcal{D}_i$ consists of experience from task i. While algorithms can choose to train the policy for task i (i.e., $\pi(\cdot|\cdot, i)$) only on $\mathcal{D}_i$, in this paper, we are interested in data-sharing schemes that correspond to relabeling data from a different task, $j \neq i$ with the reward function r_i, and learn $\pi(\cdot|\cdot, i)$ on the combined data. To be able to do so, we assume access to the functional form of the reward r_i, a common assumption in goal-conditioned RL [2, 15], and which often holds in robotics applications through the use of learned classifiers [78, 31], and discriminators [17, 8].

We assume that relabeling data $\mathcal{D}_j$ from task j to task i generates a dataset $\mathcal{D}_{j\to i}$, which is then additionally used to train on task i. Thus, the effective dataset for task i after relabeling is given by: $\mathcal{D}_i^{\text{eff}} := \mathcal{D}_i \cup (\cup_{j\neq i}\mathcal{D}_{j\to i})$. This notation simply formalizes data sharing and relabeling strategies explored in prior work [15, 32]. Our aim in this paper will be to improve on this naïve strategy, which we will show leads to significantly better results.

Offline RL algorithms. A central challenge in offline RL is distributional shift: differences between the learned policy and the behavior policy can lead to erroneous target values, where the Q-function is queried at actions $\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})$ that are far from the actions it is trained on, leading to massive overestimation [41, 35]. A number of offline RL algorithms use some kind of regularization on either the policy [35, 19, 77, 28, 63, 54] or on the learned Q-function [37, 34] to ensure that the learned policy does not deviate too far from the behavior policy. For our analysis in this work, we will abstract these algorithms into a generic policy optimization problem [37]:

$$\pi^*(\mathbf{a}|\mathbf{s}) := \arg\max_{\pi} \;\; J_{\mathcal{D}}(\pi) - \alpha D(\pi, \pi_\beta) \qquad \text{(Conservative offline RL).} \tag{1}$$

$J_{\mathcal{D}}(\pi)$ denotes the average return of policy π in the empirical MDP induced by the transitions in the dataset, and $D(\pi, \pi_\beta)$ denotes a divergence measure (e.g., KL-divergence [28, 77], MMD distance [35] or D_{CQL} [37]) between the learned policy π and the behavior policy π_β. In the multi-task offline RL setting with data-sharing, the generic optimization problem in Equation 1 for a task i utilizes the effective dataset $\mathcal{D}_i^{\text{eff}}$. In addition, we define $\pi_\beta^{\text{eff}}(\mathbf{a}|\mathbf{s}, i)$ as the effective behavior policy for task i and it is given by: $\pi_\beta^{\text{eff}}(\mathbf{a}|\mathbf{s}, i) := |\mathcal{D}_i^{\text{eff}}(\mathbf{s}, \mathbf{a})|/|\mathcal{D}_i^{\text{eff}}(\mathbf{s})|$. Hence, the counterpart of Equation 1 in the multi-task offline RL setting with data sharing is given by:

$$\forall i \in [N], \;\; \pi^*(\mathbf{a}|\mathbf{s}, i) := \arg\max_{\pi} \;\; J_{\mathcal{D}_i^{\text{eff}}}(\pi) - \alpha D(\pi, \pi_\beta^{\text{eff}}) \quad \text{(Offline RL with data sharing).} \tag{2}$$

We will utilize this generic optimization problem to motivate our method in Section 5.

4 When Does Data Sharing Actually Help in Offline Multi-Task RL?

Our goal is to leverage experience from all the tasks to learn a policy for a particular task of interest. Perhaps the simplest approach to leveraging experience across tasks is to train the task policy on not just the data coming from that task, but also relabeled data from all other tasks [5]. Is this naïve data sharing strategy sufficient for learning effective behaviors from multi-task offline data? In this section, we aim to answer this question via empirical analysis on a relatively simple domain, which is enough to reveal interesting aspects of data sharing. We first describe the experimental setup and then discuss the results and possible explanations for the observed behavior. Using insights obtained from this analysis, we will then derive a simple and effective data sharing strategy in Section 5.

Experimental analysis setup. To assess the efficacy of data sharing, we experimentally analyze various multi-task RL scenarios created with the walker2d environment in Gym [4]. We construct different test scenarios on this environment that mimic practical situations, including settings where different amounts of data of varied quality are available for different tasks [32, 79, 64]. In all these scenarios, the agent attempts three tasks: `run forward`, `run backward`, and `jump`, which we visualize in Figure 2. Following the problem statement in Section 3, these tasks share the same state-action space and transition dynamics, differing only in the reward function that the agent is trying to optimize. Different scenarios are generated via varying-size offline datasets collected via behavior policies of different qualities, *i.e.*, a single policy with mediocre or expert performance, or a mixture of policies given by the initial part of the replay buffer trained with online SAC [25] until when the final policy reaches a mediocre level of performance. We refer to these three types of offline datasets as medium, expert and medium-replay respectively following the definition in [18].

We train a single-task policy $\pi_{\text{CQL}}(\mathbf{a}|\mathbf{s}, i)$ with CQL [37] as the base offline RL method along with two forms of data-sharing, as shown in Table 1: no sharing of data across tasks (**No Sharing**)) and complete sharing of data with relabeling across all tasks (**Sharing All**), and report the performance of the resulting policy. In addition, we also measure the divergence term in Equation 2, $D(\pi(\cdot|\cdot, i), \pi_\beta^{\text{eff}}(\cdot|\cdot, i))$, for $\pi = \pi_{\text{CQL}}(\mathbf{a}|\mathbf{s}, i)$, averaged across tasks by using the Kullback-Liebler divergence. This value quantifies the average divergence between the single-task optimal policy and the relabeled behavior policy averaged across tasks.

Analysis of results in Table 1. To begin, note that even naïvely sharing data is better than not sharing any data at all on **5/9** tasks considered (compare the performance across **No Sharing** and **Sharing All** in Table 1). However, a closer look at Table 1 suggests that data-sharing can significantly degrade performance on certain tasks, especially in scenarios where the amount of data available for the original task is limited, and where the distribution of this data is narrow. For example, when using expert data for jumping in conjunction with more than 25 times as much lower-quality (mediocre & random) data for running forward and backward, we find that the agent recovers poor performance on the jumping task despite access to near-optimal jumping data.

Dataset types / Tasks	Dataset Size	Avg Return		$D_{\text{KL}}(\pi, \pi_\beta)$	
		No Sharing	**Sharing All**	**No Sharing**	**Sharing All**
medium-replay / `run forward`	109900	**998.9**	966.2	**3.70**	10.39
medium-replay / `run backward`	109980	**1298.6**	1147.5	**4.55**	12.70
medium-replay / `jump`	109511	**1603.1**	1224.7	**3.57**	15.89
average task performance	N/A	**1300.2**	1112.8	**3.94**	12.99
medium / `run forward`	27646	297.4	**848.7**	**6.53**	11.78
medium / `run backward`	31298	207.5	**600.4**	**4.44**	10.13
medium / `jump`	100000	351.1	**776.1**	**5.57**	21.27
average task performance	N/A	285.3	**747.7**	**5.51**	14.39
medium-replay / `run forward`	109900	590.1	**701.4**	**1.49**	7.76
medium / `run backward`	31298	614.7	**756.7**	**1.91**	12.2
expert / `jump`	5000	**1575.2**	885.1	**3.12**	27.5
average task performance	N/A	**926.6**	781	**2.17**	15.82

Table 1: We analyze how sharing data across all tasks (**Sharing All**) compares to **No Sharing** in the multi-task walker2d environment with three tasks: run forward, run backward, and jump. We provide three scenarios with different styles of per-task offline datasets in the leftmost column. The second column shows the number of transitions in each dataset. We report the per-task average return, the KL divergence between the single-task optimal policy and the behavior policy after the data sharing scheme, as well as the averages across tasks. **Sharing All** generally helps training while increasing the KL divergence. However, on the row highlighted in yellow, **Sharing All** yields a particualrly large KL divergence between the single-task policy and the behavior policy and degrades the performance, suggesting sharing data for all tasks is brittle.

Why does naïve data sharing degrade performance on certain tasks despite near-optimal behavior for these tasks in the original task dataset? We argue that the primary reason that naïve data sharing can actually hurt performance in such cases is because it exacerbates the distributional shift issues that afflict offline RL. Many offline RL methods combat distribution shift by implicitly or explicitly constraining the learned policy to stay close to the training data. Then, when the training data is changed by adding relabeled data from another task, the constraint causes the learned policy to change as well. When the added data is of low quality for that task, it will correspondingly lead to a lower quality learned policy for that task, unless the constraint is somehow modified. This effect is evident from the higher divergence values between the learned policy without any data-sharing and the effective behavior policy for that task *after* relabeling (e.g., expert+`jump`) in Table 1. Although these results are only for CQL, we expect that any offline RL method would, insofar as it combats distributional shift by staying close to the data, would exhibit a similar problem.

To mathematically quantify the effects of data-sharing in multi-task offline RL, we appeal to safe policy improvement bounds [39, 37, 86] and discuss cases when data-sharing between tasks i and j can degrade the amount of worst-case guaranteed improvement over the behavior policy. Prior work [37] has shown that the generic offline RL algorithm in Equation 1 enjoys the following guarantees of policy improvement on the actual MDP, beyond the behavior policy:

$$J(\pi^*) \geq J(\pi_\beta) - \mathcal{O}(1/(1-\gamma)^2)\mathbb{E}_{\mathbf{s},\mathbf{a}\sim d^\pi}\left[\sqrt{\frac{D(\pi(\cdot|\mathbf{s}), \pi_\beta(\cdot|\mathbf{s}))}{|\mathcal{D}(\mathbf{s})|}}\right] + \alpha/(1-\gamma)D(\pi, \pi_\beta). \quad (3)$$

We will use Equation 3 to understand the scenarios where data sharing can hurt. When data sharing modifies $\mathcal{D} = \mathcal{D}_i$ to $\mathcal{D} = \mathcal{D}_i^{\text{eff}}$, which includes $\mathcal{D}_i$ as a subset, it effectively aims at reducing the magnitude of the second term (i.e., sampling error) by increasing the denominator. This can be highly effective if the visitations of learned policy π^* and the dataset $\mathcal{D}$ overlap with each other in terms of state visitations. However, an increase in divergence $D(\pi(\cdot|\mathbf{s}), \pi^\beta(\cdot|\mathbf{s}))$ as a consequence of relabeling implies a potential increase in the sampling error unless the increased value of $|\mathcal{D}^{\text{eff}}(\mathbf{s})|$ precisely compensates this. Additionally, the strength of this bound also depends on the quality of the behavior data added after relabeling – if the resulting behavior policy π_β^{eff} is more suboptimal compared to π_β, i.e., $J(\pi_\beta^{\text{eff}}) < J(\pi_\beta)$, then, the guaranteed amount of improvement also reduces.

To conclude, our empirical analysis reveals that while data-sharing is often helpful in multi-task offline RL, it can lead to substantially poor performance on certain tasks as a result of exacerbated distributional shift between the optimal policy and the effective behavior policy induced after sharing data. Thus, in order to obtain effective policies for all tasks in this setting, we propose to share data *conservatively*, with precaution taken to not exacerbate distributional shift via relabeling. This forms the basis of our data-sharing scheme, CDS, which we will discuss next.

5 CDS: Reducing Distributional Shift in Multi-Task Data Sharing

The analysis in Section 4 shows that naive data sharing may be highly suboptimal in some cases, and although it often does improve over no data sharing at all in practice, it can also lead to exceedingly poor results. Can we instead devise a adaptive data sharing scheme that avoids this pathology while still retaining the benefits of data sharing?

A key insight behind our approach is to note that a data sharing scheme can be viewed as altering the dataset $\mathcal{D}_i^{\text{eff}}$, and hence the effective behavior policy, $\pi_\beta^{\text{eff}}(\mathbf{a}|\mathbf{s}, i)$. Thus, we can develop an effective data-sharing scheme by *optimizing* the objective in Equation 2 with respect to π_β^{eff} in addition to π, where π_β^{eff} belongs to the set of all possible effective behavior policies that can be obtained via any form of data sharing. We formalize this optimization below in Equation 4:

$$(\pi^*(\cdot|\cdot, i), \pi_\beta^{\text{eff}}(\cdot|\cdot, i)) := \arg\max_{\pi} \max_{\pi_\beta^{\text{eff}} \in \Pi_{\text{relabel}}} \left[J_{\mathcal{D}_i^{\text{eff}}}(\pi) - \alpha D(\pi, \pi_\beta^{\text{eff}}; i) \right], \tag{4}$$

where Π_{relabel} denotes the set of all possible behavior policies that can be obtained via relabeling.

How can we effectively convert Equation 4 into a simple condition for data sharing in multi-task offline RL? While directly solving Equation 4 is intractable since π_β^{eff} appears in both the terms and the optimal policy π cannot be expressed as a closed form function of π_β^{eff}, we can instead solve Equation 4 approximately. Fortunately, we can optimize a practical approximation to Equation 4 that uses the dataset state distribution for the policy update in Equation 4 similar to modern actor-critic methods [11, 43, 20, 25, 37]. This objective is given by: $\mathbb{E}_{\mathbf{s}\sim\mathcal{D}_i^{\text{eff}}}[\mathbb{E}_\pi[Q(\mathbf{s}, \mathbf{a}, i)] - \alpha D(\pi(\cdot|\mathbf{s}, i), \pi_\beta^{\text{eff}}(\cdot|\mathbf{s}, i))]$, which is equal to the expected "conservative Q-value" $\hat{Q}^\pi(\mathbf{s}, \mathbf{a}, i)$ on dataset states, policy actions and task i. Optimizing this objective via a co-ordinate descent on π and π_β^{eff} dictates that π be updated using a standard update of maximizing the conservative Q-function, $\hat{Q}^\pi$ (equal to the difference of the Q-function and $D(\pi, \pi_\beta^{\text{eff}}; i)$). Moreover, π_β^{eff} should also be updated towards maximizing the same expectation, $\mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}_i^{\text{eff}}}[\hat{Q}^\pi(\mathbf{s}, \mathbf{a}, i)] := \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}_i^{\text{eff}}}[Q(\mathbf{s}, \mathbf{a}, i)] - \alpha D(\pi, \pi_\beta^{\text{eff}}; i)$. This implies that when updating the behavior policy during relabeling, we should prefer state-action pairs that maximize the conservative Q-function.

Our data-sharing scheme, CDS, is based on the insight discussed above. CDS first runs any standard conservative offline RL algorithm on each task individually without any data sharing. Consistent with our insight from above, relabeling a transition that contributes a lower conservative Q-value to the current task will only decrease the average conservative Q-value of the effective resulting dataset for this task, whereas the π_β^{eff} is to be optimized to maximize this value. Hence, we employ a simple rule to decide which transitions from the data for other tasks will be relabeled to the current task, say task i: CDS checks if the conservative Q-value estimate for this transition under task i is greater than the average conservative Q-value for the original dataset for task i, $\mathcal{D}_i$. This is sufficient to prevent any degradation in the average conservative Q-value after relabeling, while still allowing us to retain the positive benefits of full data sharing. We formalize this insight in Appendix A.

Formally, assume that the task under consideration is task i. Our method CDS first estimates a conservative Q-value estimate, $\hat{Q}^\pi(\mathbf{s}, \mathbf{a}, i)$ (which can be obtained via any offline RL algorithm), and for a given transition, $(\mathbf{s}, \mathbf{a}, r_j, \mathbf{s}') \in \mathcal{D}_j$ it then checks if the following condition is satisfied:

$$\forall\, j \in [N],\ \Delta^\pi(\mathbf{s}, \mathbf{a}; j \to i) := \hat{Q}^\pi(\mathbf{s}, \mathbf{a}, i) - \mathbb{E}_{\mathbf{s}',\mathbf{a}'\sim\mathcal{D}_i}\left[\hat{Q}^\pi(\mathbf{s}', \mathbf{a}', i)\right] \geq 0. \tag{5}$$

If the condition in Equation 5 holds for the given $(\mathbf{s}, \mathbf{a})$, then the corresponding relabeled transition, $(\mathbf{s}, \mathbf{a}, r_i(\mathbf{s}, \mathbf{a}), \mathbf{s}')$ is added to $\mathcal{D}_i^{\text{eff}}$. This rule for relabeling is applied independently on each transition $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}') \in \mathcal{D}$. Finally, CDS trains a policy, $\pi(\mathbf{a}|\mathbf{s}; i)$, either conditioned on the task i (i.e., with weight sharing) or a separate $\pi(\mathbf{a}|\mathbf{s})$ policy for each task with no weight sharing, with the resulting relabeled dataset, $\mathcal{D}_i^{\text{eff}}$. Pseudocode for CDS is provided in Algorithm 1.

Algorithm 1 CDS: Conservative Data Sharing

Require: Multi-task offline dataset $\cup_{i=1}^N \mathcal{D}_i$.
1: Randomly initialize policy $\pi_\theta(\mathbf{a}|\mathbf{s}, i)$.
2: **for** $k = 1, 2, 3, \cdots$, **do**
3: Initialize $\mathcal{D}^{\text{eff}} \leftarrow \{\}$
4: **for** $i = 1, \cdots, N$ **do**
5: $\mathcal{D}_i^{\text{eff}} = \mathcal{D}_i \cup \{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_i) \in \mathcal{D}_{j\to i} : \Delta^\pi(\mathbf{s}, \mathbf{a}; j \to i) \geq 0\}$
6: Improve policy by solving eq. 2 using samples from $\mathcal{D}^{\text{eff}}$ to obtain π_θ^{k+1}.

Practical implementation details. While there are many approaches to obtain a conservative Q-function both directly and indirectly, our practical

implementation of CDS utilizes CQL for this purpose. though any base offline RL method that provides as estimate of a conservative Q-value could be used. While the theoretical version of our method utilizes a two-phased training process that first generates the relabeled dataset via single-task training and only subsequently trains on the relabeled data, we merge the steps into a single phase in practice. To do so, we use a soft transform of the condition in Equation 5 to gradually transition between single-task training and data sharing as dictated by CDS. Formally, this means for a given transition $(\mathbf{s}, \mathbf{a}, r_j, \mathbf{s}') \in \mathcal{D}_j$ under consideration to be relabeled for a task i, we weight the training objectives for the critic and the policy in our actor-critic algorithm by a soft-relabelling weight:

$$w_{\text{CDS}}(\mathbf{s}, \mathbf{a}; j \to i) := \sigma\left(\frac{\Delta(\mathbf{s}, \mathbf{a}; j \to i)}{\tau}\right), \qquad \sigma(x) = \frac{1}{1 + \exp(-x)}, \tag{6}$$

where τ is a temperature hyperparameter. This temperature parameter is chosen in a fully automatic manner utilizing adaptive temperature scaling from prior work [36] and we describe this scheme in detail in Appendix B. We also apply w_{CDS} to transitions from the same task with 0.5 probability. For the training objective of CDS for the critic and policy, see Appendix B.

6 Experimental Evaluation

We conduct experiments to answer five main questions: (1) can CDS prevent degradation in performance when sharing data as observed in Section 4?, (2) how does CDS compare to vanilla multi-task offline RL methods and prior data between tasks? (3) can CDS handle sparse reward settings, where data sharing is particularly important due to scarce supervision signal? (4) can CDS handle goal-conditioned offline RL settings where the offline dataset is undirected and highly suboptimal? (5) Is CDS able to scale to complex visual observations?

Comparisons. To answer these questions, we consider the following prior methods. On tasks with low-dimensional state spaces, we compare with the online multi-task relabeling approach **HIPI** [15], which uses inverse RL to infer for which tasks the datapoints are optimal and in practice routes a transition to task with the highest Q-value. We adapt HIPI to the offline setting by applying its data routing strategy to a conservative offline RL algorithm. We also compare to naïvely sharing data across all tasks (denoted as **Sharing All**) and vanilla multi-task offline RL method without any data sharing (denoted as **No Sharing**). On image-based domains, we compare CDS to the data sharing strategy based on human-defined skills [32] (denoted as **Skill**), which manually groups tasks into different skills (e.g. skill "pick" and skill "place") and only routes an episode to target tasks that belongs to the same skill of the source task. In these domains, we also compare to **HIPI**, **Sharing All** and **No Sharing**. We use CQL [37] as the base offline RL algorithm for all methods. For more details on the experiment set-up and hyperparameters, see Appendix B.

Multi-task environments. To address the above questions, we consider a number of multi-task reinforcement learning problems on environments visualized in Figure 2. To answer questions (1) and (2), we consider three locomotion environments from OpenAI Gym [4] with dense rewards: halfcheetah, walker2d, and ant. Each environment has three tasks, `run forward`, `run backward` and `jump`, as used in prior offline RL work [85]. To answer question (3), we also evaluate on robotic manipulation domains using environments from the Meta-World benchmark [84]. We consider four tasks: `door open`, `door close`, `drawer open` and `drawer close`. Meaningful data sharing requires a consistent state representation across tasks, so we put both the door and the drawer on the same table, as shown in Figure 2. Each task has a sparse reward of 1 when the success condition is met and 0 otherwise. To answer question (4), we consider maze navigation tasks where the temporal "stitching" ability of an offline RL algorithm is crucial to obtain good performance. We create goal reaching tasks using the ant robot in the medium and hard mazes from D4RL [18]. The set of goals is a fixed discrete set of size 7 and 3 for large and medium mazes, respectively. Following Fu et al. [18], a reward of +1 is given and the episode terminates if the state is within a threshold radius of the goal. Finally, to explore how CDS scales to image-based manipulation tasks (question (5)), we utilize a simulation environment similar to the real-world setup presented in [32]. This environment, which was utilized by Kalashnikov et al. [32] as a representative and realistic simulation of a real-world robotic manipulation problem, consists of 10 image-based

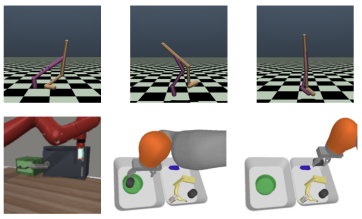

Figure 2: Our environments (from left to right): walker2d `run forward`, walker2d `run backward`, walker2d `jump`, Meta-World `door open/close` and `drawer open/close` and vision-based picking and placing tasks in [32].

Environment	Tasks / Dataset type	CDS (ours)	HIPI [15]	Sharing All	No Sharing
walker2d	run forward / medium-replay	**1100.7**	692.2	701.4	590.1
	run backward / medium	638.4	664.9	**756.7**	614.7
	jump / expert	1538.4	**1604.4**	885.1	1575.2
	average	**1092.5**	987.2	781	926.6
Meta-World [84]	door open / medium-replay	**57.3%**	30.7%	19.1%	0%
	door close / expert	**33.3%**	0%	21%	2%
	drawer open / expert	76%	45.3%	**85.5%**	0%
	drawer close / medium-replay	99.7%	66.7%	**100%**	5.7%
	average	**66.6%**	35.7%	56.4%	1.9%
AntMaze [18]	large maze (7 tasks) / undirected	**0.23**	0.01	0.17	0.13
	large maze (7 tasks) / directed	**0.24**	0.12	0.21	0.23
	medium maze (3 tasks) / undirected	**0.37**	0.07	0.23	0.22
	medium maze (3 tasks) / directed	**0.19**	0.08	0.12	0.17

Table 2: Results for multi-task locomotion (walker2d), robotic manipulation (Meta-World) and navigation environments (AntMaze) with low-dimensional state inputs. We include per-task performance for walker2d and Meta-World domains and the overall performance averaged across tasks (highlighted in gray) for all three domains. We bold the highest score across all methods. CDS performs achieves the best or comparable performance on all of these multi-task environments.

manipulation tasks that involve different combinations of picking specific objects (banana, bottle, sausage, milk box, food box, can and carrot) and placing them in one of the three fixtures (bowl, plate and divider plate) (see example task images in Fig. 2). We pick these tasks due to their similarity to the real-world setup introduced by Kalashnikov et al. [32], which utilized a skill-based data-sharing heuristic strategy (**Skill**) for data-sharing that significantly outperformed simple data-sharing alternatives, which we use as a point of comparison. More details on the environments are in the appendix. We report the average return for locomotion tasks, normalized score as proposed in [18] for AntMaze, and success rate both manipluation environments, averaged over 3 random seeds.

Multi-task datasets. Following the analysis in Section 4, we intentionally construct datasets with a variety of heterogeneous behavior policies to test if CDS can provide effective data sharing to improve performance while avoiding harmful data sharing that exacerbates distributional shift. For the locomotion domains, we use a large, diverse dataset (medium-replay) for `run forward`, a medium-sized dataset for `run backward`, and an expert dataset with limited data for `run jump`. For Meta-World, we consider medium-replay datasets with 152K transitions for task `door open` and `drawer close` and expert datasets with only 2K transitions for task `door close` and `drawer open`. For AntMaze, we modify the D4RL datasets for antmaze-*-play environments to construct two kinds of multi-task datasets: an "undirected" dataset, where data is equally divided between different tasks and the rewards are correspondingly relabeled, and a "directed" dataset, where a trajectory is associated with the goal closest to the final state of the trajectory. This means that the per-task data in the undirected setting may not be relevant to reaching the goal of interest. Thus, data-sharing is crucial for good performance: methods that do not effectively perform data sharing and train on largely task-irrelevant data are expected to perform worse. Finally, for the image-based manipulation tasks, we collect datasets for all the tasks individually by running online RL [31] until the task reaches medium-level performance (40% for picking tasks and 80% placing tasks). At that point, we merge the entire replay buffers from different tasks creating a final dataset of 100K RL episodes where each episode consists of 25 transitions.

Results on domains with low-dimensional states. We present the results on all non-vision environments in Table 2, but leave the results of halfcheetah and ant to Appendix C. CDS achieves the best average performance across all environments. On the locomotion domain, we observe the most significant improvement on task `jump` on all three environments. We interpret this as strength of conservative data sharing, which mitigates the distribution shift that can be introduced by routing large amount of other task data to the task with limited data and narrow distribution. We also validate this by measuring the $D_{\mathrm{KL}}(\pi, \pi_\beta)$ in Table 3 where

Environment	Dataset types / Tasks	$D_{\mathrm{KL}}(\pi, \pi_\beta)$ No Sharing	Sharing All	CDS (ours)
walker2d	medium-replay / run forward	**1.49**	7.76	**1.49**
	medium / run backward	**1.91**	12.2	6.09
	expert / jump	3.12	27.5	**2.91**

Table 3: Measuring $D_{\mathrm{KL}}(\pi, \pi_\beta)$ on the walker2d environment. **Sharing All** degrades the performance on task jump with limited expert data as discussed in Table 1. CDS manages to obtain a behavior policy after data sharing that is closer to the single-task optimal policy in terms of the KL divergence compared to **No Sharing** and **Sharing All** on task `jump` (highlighted in yellow). Since CDS also achieves better performance, this analysis suggests that reducing distribution shift is important for effective offline data sharing.

Task Name	CDS (ours)	HIPI [15]	Skill [32]	Sharing All	No Sharing
lift-banana	**54.0%**	39.7%	33.6%	45.6%	12.6%
lift-bottle	**76.3%**	58.5%	53.3%	42.8%	44.5%
lift-sausage	**75.9%**	65.6%	62.5%	73.8%	55.2%
lift-milk	**82.7%**	75.3%	62.8%	68.9%	58.9%
lift-food	**70.3%**	64.6%	23.1%	64.9%	29.5%
lift-can	**76.1%**	70.8%	37.6%	49.4%	41.8%
lift-carrot	**80.4%**	70.1%	69.4%	72.2%	63.1%
place-bowl	84.4%	72.0%	**84.5%**	64.7%	77.0%
place-plate	**86.9%**	82.2%	79.5%	75.1%	82.2%
place-divider-plate	**89.4%**	72.6%	81.0%	79.3%	84.7%
average	**77.6%**	67.2%	58.7%	63.7%	55.0%

Table 4: Results for multi-task vision-based robotic manipulation domains in [32]. We consider 7 tasks denoted as `lift-object` where the goal of each task is to lift a different object and 3 tasks denoted as `place-fixture` where the goal of each task is to place a lifted object onto different fixtures. The numbers are in success rates. CDS outperforms both a skill-based data sharing strategy [32] (**Skill**) and other data sharing methods on the average task success rate (highlighted in gray) and 9 out of 10 per-task success rate.

π_β is the behavior policy after we perform CDS to share data. As shown in Table 3, CDS achieves lower KL divergence between the single-task optimal policy and the behavior policy after data sharing on task `jump` with limited expert data, whereas **Sharing All** results in much higher KL divergence compared to **No Sharing** as discussed in Section 4 and Table 1. Hence, CDS is able to mitigate distribution shift when sharing data and result in performance boost.

On the Meta-World tasks, we find that the agent without data sharing completely fails to solve most of the tasks due to the low quality of the medium replay datasets and the insufficient data for the expert datasets. **Sharing All** improves performance since in the sparse reward settings, data sharing can introduce more supervision signal and help training. CDS further improves over **Sharing All**, suggesting that CDS can not only prevent harmful data sharing, but also lead to more effective multi-task learning compared to **Sharing All** in scenarios where data sharing is imperative.

In the AntMaze tasks, we observe that CDS performs better than **Sharing All** and drastically outperforms HIPI in all four settings. Perhaps surprisingly, **No Sharing** is a strong baseline, however, is significantly outperformed by CDS in the harder setting with undirected data. Moreover, CDS performs on-par or better in the undirected setting compared to the directed setting, indicating the effectiveness of CDS in routing data in challenging settings.

Results on image-based robotic manipulation domains. Here, we compare CDS to the hand-designed **Skill** sharing strategy, in addition to the other methods. Since CDS is applicable to any offline multi-task RL algorithm, we employ it as a separate data-sharing strategy in [32] while keeping the model architecture and all the other hyperparameters constant, which allows us to carefully evaluate the influence of data sharing in isolation. The results are reported in Table 4. CDS outperforms both **Skill** and other approaches, indicating that CDS is able to scale to high-dimensional observation inputs and can effectively remove the need for manual curation of data sharing strategies.

7 Conclusion

In this paper, we study the multi-task offline RL setting, focusing on the problem of sharing offline data across tasks for better multi-task learning. Through empirical analysis, we identify that naïvely sharing data across tasks generally helps learning but can significantly hurt performance in scenarios where excessive distribution shift is introduced. To address this challenge, we present conservative data sharing (CDS), which relabels data to a task when the conservative Q-value of the given transition is better than the expected conservative Q-value of the target task. On multitask locomotion, manipulation, navigation, and vision-based manipulation domains, CDS consistently outperforms or achieves comparable performance to existing data sharing approaches. While CDS attains superior results, it is not able to handle data sharing in settings where dynamics vary across tasks. Another limitation is that CDS requires functional forms of rewards. We leave these as future work.

Finally, regarding negative societal impact, one potential risk the reward design in RL which is challenging to obtain in the real world. For example, it is generally infeasible to specify rewards for (offline) RL in real-world applications, making (offline) RL hard to scale to real robots. Moreover, training procedures of ML models are generally compute-intensive and collecting large datasets for offline RL is also burdensome, which cause inequitable access to these models and datasets. We would like push RL community to address these risks in the near future.

References

[1] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pages 104–114. PMLR, 2020.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.

[3] Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. *arXiv preprint arXiv:2008.05556*, 2020.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[5] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[6] Denis Charles, Max Chickering, and Patrice Simard. Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research*, 14, 2013.

[7] Yevgen Chebotar, Karol Hausman, Yao Lu, Ted Xiao, Dmitry Kalashnikov, Jake Varley, Alex Irpan, Benjamin Eysenbach, Ryan Julian, Chelsea Finn, and Sergey Levine. Actionable models: Unsupervised offline reinforcement learning of robotic skills. *arXiv preprint arXiv:2104.07749*, 2021.

[8] Annie S Chen, Suraj Nair, and Chelsea Finn. Learning generalizable robotic reward functions from" in-the-wild" human videos. *arXiv preprint arXiv:2103.16817*, 2021.

[9] Paul Christiano, Jan Leike, Tom B Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *arXiv preprint arXiv:1706.03741*, 2017.

[10] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning, 2020.

[11] Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*, 2012.

[12] Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. Sharing knowledge in multi-task deep reinforcement learning. In *International Conference on Learning Representations*, 2019.

[13] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

[14] Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, 2018.

[15] Benjamin Eysenbach, Xinyang Geng, Sergey Levine, and Ruslan Salakhutdinov. Rewriting history with inverse rl: Hindsight inference for policy improvement. *arXiv preprint arXiv:2002.11089*, 2020.

[16] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

[17] Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. Variational inverse control with events: A general framework for data-driven reward definition. *arXiv preprint arXiv:1805.11686*, 2018.

[18] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

[19] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900*, 2018.

[20] Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[21] Florent Garcin, Boi Faltings, Olivier Donatsch, Ayar Alazzawi, Christophe Bruttin, and Amr Huber. Offline and online evaluation of news recommender systems at swissinfo. ch. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 169–176, 2014.

[22] Dibya Ghosh, Avi Singh, Aravind Rajeswaran, Vikash Kumar, and Sergey Levine. Divide-and-conquer reinforcement learning. *arXiv preprint arXiv:1711.09874*, 2017.

[23] Arthur Guez, Robert D Vincent, Massimo Avoli, and Joelle Pineau. Adaptive treatment of epilepsy via batch-mode reinforcement learning. In *AAAI*, pages 1671–1678, 2008.

[24] 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.

[25] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

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

[27] Zhiao Huang, Fangchen Liu, and Hao Su. Mapping state space using landmarks for universal goal reaching. *Advances in Neural Information Processing Systems*, 32:1942–1952, 2019.

[28] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

[29] Natasha Jaques, Judy Hanwen Shen, Asma Ghandeharioun, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Shane Gu, and Rosalind Picard. Human-centric dialog training via offline reinforcement learning. *arXiv preprint arXiv:2010.05848*, 2020.

[30] Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, pages 1094–1099. Citeseer, 1993.

[31] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673. PMLR, 2018.

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

[33] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *arXiv preprint arXiv:2005.05951*, 2020.

[34] Ilya Kostrikov, Jonathan Tompson, Rob Fergus, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. *arXiv preprint arXiv:2103.08050*, 2021.

[35] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pages 11761–11771, 2019.

[36] Aviral Kumar, Abhishek Gupta, and Sergey Levine. Discor: Corrective feedback in reinforcement learning via distribution correction. *arXiv preprint arXiv:2003.07305*, 2020.

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

[38] Sascha Lange, Thomas Gabel, and Martin A. Riedmiller. Batch reinforcement learning. In *Reinforcement Learning*, volume 12. Springer, 2012.

[39] Romain Laroche, Paul Trichelair, and Remi Tachet Des Combes. Safe policy improvement with baseline bootstrapping. In *International Conference on Machine Learning*, pages 3652–3661. PMLR, 2019.

[40] Byung-Jun Lee, Jongmin Lee, and Kee-Eung Kim. Representation balancing offline model-based reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=QpNz8r_Ri2Y.

[41] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[42] Alexander C Li, Lerrel Pinto, and Pieter Abbeel. Generalized hindsight for reinforcement learning. *arXiv preprint arXiv:2002.11708*, 2020.

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

[44] Xingyu Lin, Harjatin Singh Baweja, and David Held. Reinforcement learning without ground-truth state. *arXiv preprint arXiv:1905.07866*, 2019.

[45] Hao Liu, Alexander Trott, Richard Socher, and Caiming Xiong. Competitive experience replay. *arXiv preprint arXiv:1902.00528*, 2019.

[46] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Off-policy policy gradient with state distribution correction. *CoRR*, abs/1904.08473, 2019.

[47] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Provably good batch reinforcement learning without great exploration. *arXiv preprint arXiv:2007.08202*, 2020.

[48] Corey Lynch and Pierre Sermanet. Grounding language in play. *arXiv preprint arXiv:2005.07648*, 2020.

[49] Ajay Mandlekar, Fabio Ramos, Byron Boots, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Dieter Fox. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4414–4420. IEEE, 2020.

[50] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. *arXiv preprint arXiv:2006.03647*, 2020.

[51] Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

[52] Ashvin Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *arXiv preprint arXiv:1807.04742*, 2018.

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

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

[55] Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. Temporal difference models: Model-free deep rl for model-based control. *arXiv preprint arXiv:1802.09081*, 2018.

[56] Doina Precup, Richard S Sutton, and Sanjoy Dasgupta. Off-policy temporal-difference learning with function approximation. In *ICML*, pages 417–424, 2001.

[57] Rafael Rafailov, Tianhe Yu, A. Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. *Learning for Decision Making and Control (L4DC)*, 2021.

[58] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer, 2005.

[59] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

[60] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR, 2015.

[61] Tom Schaul, Diana Borsa, Joseph Modayil, and Razvan Pascanu. Ray interference: a source of plateaus in deep reinforcement learning. *arXiv preprint arXiv:1904.11455*, 2019.

[62] Susan M Shortreed, Eric Laber, Daniel J Lizotte, T Scott Stroup, Joelle Pineau, and Susan A Murphy. Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Machine learning*, 84(1-2):109–136, 2011.

[63] 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. *arXiv preprint arXiv:2002.08396*, 2020.

[64] Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. *arXiv preprint arXiv:2011.10024*, 2020.

[65] Avi Singh, Albert Yu, Jonathan Yang, Jesse Zhang, Aviral Kumar, and Sergey Levine. Cog: Connecting new skills to past experience with offline reinforcement learning. *arXiv preprint arXiv:2010.14500*, 2020.

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

[67] Alex Strehl, John Langford, Sham Kakade, and Lihong Li. Learning from logged implicit exploration data. *arXiv preprint arXiv:1003.0120*, 2010.

[68] Hao Sun, Zhizhong Li, Xiaotong Liu, Dahua Lin, and Bolei Zhou. Policy continuation with hindsight inverse dynamics. *arXiv preprint arXiv:1910.14055*, 2019.

[69] Richard S Sutton, A Rupam Mahmood, and Martha White. An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research*, 17(1):2603–2631, 2016.

[70] Adith Swaminathan and Thorsten Joachims. Batch learning from logged bandit feedback through counterfactual risk minimization. *J. Mach. Learn. Res*, 16:1731–1755, 2015.

[71] Phillip Swazinna, Steffen Udluft, and Thomas Runkler. Overcoming model bias for robust offline deep reinforcement learning. *arXiv preprint arXiv:2008.05533*, 2020.

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

[73] Georgios Theocharous, Philip S Thomas, and Mohammad Ghavamzadeh. Ad recommendation systems for life-time value optimization. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1305–1310, 2015.

[74] Philip S Thomas, Georgios Theocharous, Mohammad Ghavamzadeh, Ishan Durugkar, and Emma Brunskill. Predictive off-policy policy evaluation for nonstationary decision problems, with applications to digital marketing. In *AAAI*, pages 4740–4745, 2017.

[75] L. Wang, Wei Zhang, Xiaofeng He, and H. Zha. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

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

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

[78] Annie Xie, Avi Singh, Sergey Levine, and Chelsea Finn. Few-shot goal inference for visuomotor learning and planning. In *Conference on Robot Learning*, pages 40–52. PMLR, 2018.

[79] Annie Xie, Frederik Ebert, Sergey Levine, and Chelsea Finn. Improvisation through physical understanding: Using novel objects as tools with visual foresight. *Robotics: Science and Systems (RSS)*, 2019.

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

[81] Rui Yang, Jiafei Lyu, Yu Yang, Jiangpeng Ya, Feng Luo, Dijun Luo, Lanqing Li, and Xiu Li. Bias-reduced multi-step hindsight experience replay. *arXiv preprint arXiv:2102.12962*, 2021.

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

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

[84] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pages 1094–1100. PMLR, 2020.

[85] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239*, 2020.

[86] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *arXiv preprint arXiv:2102.08363*, 2021.

[87] Wenxuan Zhou, Sujay Bajracharya, and David Held. Plas: Latent action space for offline reinforcement learning. *arXiv preprint arXiv:2011.07213*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Please see Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Please see Section 7.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]

(b) Did you include complete proofs of all theoretical results? [Yes] Proofs can be found in Appendix A.

3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We provide code, data and instructions for Gym and Antmaze experiments and code/data for all experiments will be made public with the final.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Please see Appendix B

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Please see Appendix B for the full results with error bars.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please see Appendix B

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] Yes we cite Fu et al. [18], Yu et al. [84], Kalashnikov et al. [32].

(b) Did you mention the license of the assets? [Yes] Please see Appendix B

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We provide datasets for some of our tasks with the supplementary material.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] Datasets in [18, 84] are open-sourced and we corresponded with the authors of [32] to access the dataset.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] There is no personally identifiable information or offensive content, since the tasks are simulated RL tasks.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]