Multi-Task Offline Reinforcement Learning with Conservative Data Sharing

Tianhe Yu [*1,2] **Aviral Kumar** [*3] **Yevgen Chebotar** [2] **Karol Hausman** [2] **Sergey Levine** [3] **Chelsea Finn** [1,2]

Abstract

Many recent offline RL algorithms attain both good empirical performance and enjoy theoretical guarantees, but their applicability is limited to settings where data is collected for only solving a single task. However, a natural use case of such methods 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 which is expected to improved performance due to better handling of sampling error, 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. We characterize the tradeoff and devise a simple technique for data-sharing in multi-task offline RL, conservative data sharing (CDS) that we theoretically analyze. Empirically, CDS attains outperforms prior methods on challenging problems including locomotion, maze navigation and real-world robotic manipulation domains.

1. Introduction

Recent advances in offline reinforcement learning (RL) make it possible to train policies for real-world scenarios, such as robotics [32, 59, 33] and healthcare [24, 65], 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. Indeed, many existing datasets in robotics [17, 11] and offline RL [19] 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 [33], resulting in a brittle method with unpredictable results in practice. In this paper, we aim to theoretically and empirically understand how data sharing affects RL performance specifically in the offline setting, and develop an effective method for selectively sharing data across tasks.

Similar to prior works in the *online* RL setting with exploration that have noted that multi-task training can often lead to worse performance than training on each task individually [54, 61, 87], we multi-task RL remains a challenging problem even in the offline setting, particularly when sharing data across tasks in the *absence of exploration*. While prior works have developed heuristic methods for reweighting and relabeling data [3, 16, 43, 33], 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, utilizing the tools from the safe policy improvement framework [40, 56, 38, 89], we show 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 theoretically, and present empirical 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 while also retaining the reduced sampling error as a result of increased dataset size. We then derive a practical instantiation of this procedure by optimizing a lower bound on the training objective that selectively relabels 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 con-

[*]Equal contribution [1]Stanford University [2]Robotics at Google [3]UC Berkeley. Correspondence to: Tianhe Yu <tianheyu@cs.stanford.edu>, Aviral Kumar <aviralk@berkeley.edu>.

Accepted at *ICML 2021 Workshop on Reinforcement Learning Theory*.

servative data sharing (CDS), is able to prevent performance degradation due to excessive data sharing, while still providing large gains over naïve single-task training. Via extensive evaluations on a number of maze navigation, locomotion and robotic manipulation domains, we find that CDS is the only method to attain good performance across the board, often significantly outperforming the best *domain-specific* baseline, improving over the next best baseline on each domain by **17.5%** on average. On the other hand, no single baseline approach performs reasonably on all the domains.

2. 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 [19], distinct object manipulation objectives [81], and different user preferences [10]. 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 $\mathbf{s}$ 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 [3, 16], and which often holds in robotics applications through the use of learned rewards [81, 32, 18, 9].

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 [16, 33]. 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 [42, 36]. A number of offline RL algorithms use some kind of regularization on either the policy [36, 20, 80, 29, 66, 55] or on the learned Q-function [38, 35] 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 [38]:

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

$J_\mathcal{D}(\pi)$ 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 [29, 80], MMD distance [36] or D_{CQL} [38]) between the learned policy π and the behavior policy π_β, where $D_{\text{CQL}}(p, q)$ denote the following distance between two distributions $p(\mathbf{x})$ and $q(\mathbf{x})$ with equal support $\mathcal{X}$:

$$D_{\text{CQL}}(p, q) := \sum_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}) \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} - 1\right).$$

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:

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

We will utilize this generic optimization problem to motivate our method in Section 4.

3. Characterizing When Data Sharing Actually Helps 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. 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 [6]. 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 observe that data sharing degrades performance when adding low quality data leads to a lower quality policy and exacerbates distributional shift in offline RL using empirical analysis detailed in Appendix C. We mathematically analyze this issue and will then derive a simple and effective data sharing strategy in Section 4.

To formally characterize the effects of data-sharing in multi-task offline RL, we appeal to safe policy improvement bounds [40, 38, 89] 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 [38] 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:

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

In the above equation, $(J_\mathcal{D}(\pi) - J_\mathcal{D}(\pi_\beta))$ denotes the improvement of π over π_β in the empirical MDP induced by the dataset, which Equation 1 optimizes directly. However, we incur a prince due to sampling error that grows quadratically in the horizon $1/(1-\gamma)^2$. 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 as in the case of naïve data sharing, it effectively aims at reducing the magnitude of the second term (i.e., sampling error) by increasing the denominator $|\mathcal{D}(\mathbf{s})|$. This can be highly effective if the 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.

Practical implications. As we show in Appendix C (Table 2), we find that the conclusions drawn above match practical scenarios, and in cases where data sharing leads to poor performance, it does so when indeed distributional shift is increased. This implies that while data-sharing can often be 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.

4. CDS: Reducing Distributional Shift in Multi-Task Data Sharing

The analysis in Section 3 shows that naïve 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:

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

where Π_{relabel} denotes the set of all possible behavior policies that can be obtained via relabeling. The next result characterizes safe policy improvement for Equation 4 and discusses how it leads to improvement over the behavior policy and also produces an effective practical method.

Proposition 4.1 (Characterizing safe-policy improvement for CDS.)**.** *Let $\pi^*(\mathbf{a}|\mathbf{s})$ be the policy obtained by optimizing Equation 4, and let $\pi_\beta(\mathbf{a}|\mathbf{s})$ be the behavior policy for $\mathcal{D}_i$. Then, w.h.p. $\geq 1-\delta$, π^* is a ζ-safe policy improvement over π_β, i.e., $J(\pi^*) \geq J(\pi_\beta) - \zeta$, where ζ is given by:*

$$\zeta = \mathcal{O}\left(\frac{1}{(1-\gamma)^2}\right)\mathbb{E}_{\mathbf{s}\sim d^{\pi^*}_{\mathcal{D}_i^{\text{eff}}}}\left[\sqrt{\frac{D_{CQL}(\pi^*, \pi^*_\beta)(\mathbf{s}) + 1}{|\mathcal{D}_i^{\text{eff}}(\mathbf{s})|}}\right] \\ - \left[\alpha D(\pi^*, \pi^*_\beta) + \underbrace{J(\pi^*_\beta) - J(\pi_\beta)}_{(a)}\right],$$

where $\mathcal{D}_i^{\text{eff}} \sim d^{\pi^*_\beta}(\mathbf{s})$ and $\pi^*_\beta(\mathbf{a}|\mathbf{s})$ denotes the policy $\pi \in \Pi_{\text{relabel}}$ that maximizes Equation 4.

A proof and analysis of this proposition is provided in Appendix A. Using simple arguments, we note that the bound in Proposition 4.1 is stronger than both no data sharing as well as naïve data sharing. We show in Appendix A that optimizing Equation 4 reduces the numerator $D_{\text{CQL}}(\pi^*, \pi^*_\beta)$ term while also increasing $|\mathcal{D}_i^{\text{eff}}(\mathbf{s})|$, thus reducing the amount of sampling error. In addition, Lemma A.1 shows that the improvement term (a) is guaranteed to be positive if a large enough α is chosen in Equation 4. Combining these,

Task Name	CDS (ours)	HIPI [16]	Skill [33]	Sharing All	No Sharing
average of 10 lift and place tasks	**77.6%**	67.2%	58.7%	63.7%	55.0%

Table 1. Results for multi-task vision-based robotic manipulation domains in [33]. CDS outperforms prior methods on the average task success rate. **HIPI** [16] relabels a given transition to a particular task, **Skill** [33] is a method that uses a "hand-designed" data sharing scheme, and **Sharing All** refers to sharing all the data, while **No Sharing** refers to not performing any data sharing.

we find data sharing using Equation 4 improves over both complete data sharing (which may increase $D_{\text{CQL}}(\pi, \pi_\beta)$) and no data sharing (which does not increase $|\mathcal{D}_i^{\text{eff}}(\mathbf{s})|$).

A practical algorithm, CDS. The next step is to effectively convert Equation 4 into a simple condition for data sharing in multi-task offline RL. While directly solving Equation 4 is intractable in practice, since both the terms depend on $\pi_\beta^{\text{eff}}(\mathbf{a}|\mathbf{s})$, and somewhat non-trivially (since the first term $J_{\mathcal{D}^{\text{eff}}}(\pi)$ depends on the empirical MDP induced by the effective behavior policy), we need to instead solve Equation 4 approximately. Fortunately, we can optimize a *lower-bound approximation* to Equation 4 that uses the dataset state distribution for the policy update in Equation 4 similar to modern actor-critic methods [12, 44, 21, 26, 38] which only introduces an additional $D(\pi, \pi_\beta)$ term in the objective. 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. Thus, 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:

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

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. For practical implementation details of CDS see Appendix E.

Figure 1. Visualization of the vision-based picking and placing tasks in [33] evaluated in Table 1.

Experimental evaluation of CDS. We present an extensive empirical evaluation of CDS in Appendix D, but we present one set of empirical results briefly here. We evaluate CDS on a multi-task vision-based robotic manipulation domain proposed in [33], which consists of 7 lifting tasks and 3 placing tasks (visualized in Figure 1). We compare CDS to prior methods (details in Appendix D) and show that CDS outperforms all the previous methods in the average success rate across 10 tasks, suggesting the effectiveness of CDS.

5. 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. We first 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. We provide theoretical analysis of CDS and also show that CDS outperforms prior approaches in a range of multi-task problems.

References

[1] Abbas Abdolmaleki, Jost Tobias Springenberg, Y. Tassa, R. Munos, N. Heess, and Martin A. Riedmiller. Maximum a posteriori policy optimisation. *ArXiv*, abs/1806.06920, 2018.

[2] 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.

[3] 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.

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

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

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

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

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

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

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

[33] 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.

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

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

[36] 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.

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

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

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

[40] 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.

[41] 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.

[42] 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.

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

[44] 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.

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

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

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

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

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

[50] 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.

[51] 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.

[52] 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.

[53] 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.

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

[55] 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.

[56] Marek Petrik, Yinlam Chow, and Mohammad Ghavamzadeh. Safe policy improvement by minimizing robust baseline regret. *arXiv preprint arXiv:1607.03842*, 2016.

[57] 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.

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

[59] 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.

[60] 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.

[61] 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.

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

[63] 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.

[64] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[65] 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.

[66] 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.

[67] 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.

[68] 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.

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

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

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

[72] 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.

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

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

[75] 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.

[76] 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.

[77] 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.

[78] 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.

[79] 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.

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

[81] 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.

[82] 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.

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

[84] 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.

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

[86] 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.

[87] 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.

[88] 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.

[89] 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.

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

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}.

A. Analysis of CDS

In this section, we will analyze the key idea behind our method CDS (Section 4) and show that the abstract version of our method (Equation 4) provides better policy improvement guarantees than naïve data sharing and that the practical version of our method (Equation 5) approximates Equation 4 resulting in an effective practical algorithm.

A.1. Analysis of the Algorithm in Equation 4

We begin with analyzing Equation 4, which is used to derive the practical variant of our method, CDS. We build on the analysis of safe-policy improvement guarantees of conventional offline RL algorithms [40, 38] and show that data sharing using CDS attains better guarantees in the worst case. To begin the analysis, we introduce some notation and prior results that we will directly compare to.

Notation and prior results. Let $\pi_\beta(\mathbf{a}|\mathbf{s})$ denote the behavior policy for task i (note that index i was dropped from $\pi_\beta(\mathbf{a}|\mathbf{s}; i)$ for brevity). The dataset, $\mathcal{D}_i$ is generated from the marginal state-action distribution of π_β, i.e., $\mathcal{D} \sim d^{\pi_\beta}(\mathbf{s})\pi_\beta(\mathbf{a}|\mathbf{s})$. We define $d^{\pi}_{\mathcal{D}}$ as the state marginal distribution introduced by the dataset $\mathcal{D}$ under π. Let $D_{\text{CQL}}(p, q)$ denote the following distance between two distributions $p(\mathbf{x})$ and $q(\mathbf{x})$ with equal support $\mathcal{X}$:

$$D_{\text{CQL}}(p, q) := \sum_{\mathbf{x} \in \mathcal{X}} p(\mathbf{x}) \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} - 1 \right).$$

Unless otherwise mentioned, we will drop the subscript "CQL" from D_{CQL} and use D and D_{CQL} interchangeably. Prior works [38] have shown that the optimal policy π_i^* that optimizes Equation 1 attains a high probability safe-policy improvement guarantee, i.e., $J(\pi_i^*) \geq J(\pi_\beta) - \zeta_i$, where ζ_i is:

$$\zeta_i = \mathcal{O}\left(\frac{1}{(1-\gamma)^2} \right) \mathbb{E}_{\mathbf{s} \sim d^{\pi_i^*}_{\mathcal{D}_i}} \left[\sqrt{\frac{D_{\text{CQL}}(\pi_i^*, \pi_\beta)(\mathbf{s}) + 1}{|\mathcal{D}_i(\mathbf{s})|}} \right] + \alpha D(\pi_i^*, \pi_\beta). \tag{6}$$

The first term in Equation 6 corresponds to the decrease in performance due to sampling error and this term is high when the single-task optimal policy π_i^* visits rarely observed states in the dataset $\mathcal{D}_i$ and/or when the divergence from the behavior policy π_β is higher under the states visited by the single-task policy $\mathbf{s} \sim d^{\pi_i^*}_{\mathcal{D}_i}$.

Let $J_{\mathcal{D}}(\pi)$ denote the return of a policy π in the empirical MDP induced by the transitions in the dataset $\mathcal{D}$. Further, let us assume that optimizing Equation 4 gives us the following policies:

$$\pi^*(\mathbf{a}|\mathbf{s}), \pi^*_\beta(\mathbf{a}|\mathbf{s}) := \arg \max_{\pi, \pi_\beta \in \Pi_{\text{relabel}}} \underbrace{J_{\mathcal{D}_i^{\text{eff}}}(\pi) - \alpha D(\pi, \pi_\beta)}_{:= f(\pi, \pi_\beta; \mathcal{D}_i^{\text{eff}})}, \tag{7}$$

where the optimized behavior policy π^*_β is constrained to lie in a set of all policies that can be obtained via relabeling, Π_{relabel}, and the dataset, $\mathcal{D}_i^{\text{eff}}$ is sampled according to the state-action marginal distribution of π^*_β, i.e., $\mathcal{D}_i^{\text{eff}} \sim d^{\pi^*_\beta}(\mathbf{s}, \mathbf{a})$. Additionally, for convenience, define, $f(\pi_1, \pi_2; \mathcal{D}) := J_{\mathcal{D}}(\pi_1) - \alpha D(\pi_1, \pi_2)$ for any two policies π_1 and π_2, and a given dataset $\mathcal{D}$.

We now show the following result for CDS:

Proposition A.1 (Proposition 4.1 restated)**.** *Let $\pi^*(\mathbf{a}|\mathbf{s})$ be the policy obtained by optimizing Equation 4, and let $\pi_\beta(\mathbf{a}|\mathbf{s})$ be the behavior policy for $\mathcal{D}_i$. Then, w.h.p. $\geq 1 - \delta$, π^* is a ζ-safe policy improvement over π_β, i.e., $J(\pi^*) \geq J(\pi_\beta) - \zeta$,*

where ζ is given by:

$$\zeta = \mathcal{O}\left(\frac{1}{(1-\gamma)^2}\right)\mathbb{E}_{\mathbf{s}\sim d^{\pi^*}_{\mathcal{D}_i^{\text{eff}}}}\left[\sqrt{\frac{D_{CQL}(\pi^*, \pi^*_\beta)(\mathbf{s})+1}{|\mathcal{D}_i^{\text{eff}}(\mathbf{s})|}}\right] - \left[\alpha D(\pi^*, \pi^*_\beta) + \underbrace{J(\pi^*_\beta) - J(\pi_\beta)}_{(a)}\right],$$

where $\mathcal{D}_i^{\text{eff}} \sim d^{\pi^*_\beta}(\mathbf{s})$ and $\pi^*_\beta(\mathbf{a}|\mathbf{s})$ denotes the policy $\pi \in \Pi_{relabel}$ that maximizes Equation 4.

Proof. To prove this proposition, we shall quantify the lower-bound on the improvement in the policy performance due to Equation 7 in the empirical MDP, and the potential drop in policy performance in the original MDP due to sampling error, and combine the terms to obtain our bound. First note that for any given policy π, and a dataset $\mathcal{D}_i^{\text{eff}}$ with effective behavior policy $\pi_\beta(\mathbf{a}|\mathbf{s})$, the following bound holds [38]:

$$J(\pi) \geq J_{\mathcal{D}_i^{\text{eff}}}(\pi) - \mathcal{O}\left(\frac{1}{(1-\gamma)^2}\right)\mathbb{E}_{\mathbf{s}\sim d^{\pi}_{\mathcal{D}_i^{\text{eff}}}}\left[\sqrt{\frac{D_{\text{CQL}}(\pi, \pi^*_\beta)(\mathbf{s})+1}{|\mathcal{D}_i^{\text{eff}}(\mathbf{s})|}}\right], \tag{8}$$

where the $\mathcal{O}(\cdot)$ notation hides constants depending upon the concentration properties of the MDP [40] and $1-\delta$, the probability with which the statement holds. Next, we provide guarantees on policy improvement in the empirical MDP. To see this, note that the following statements on $f(\pi_1, \pi_2; \mathcal{D})$ are true:

$$\forall \pi' \in \Pi_{\text{relabel}}, \quad f(\pi^*, \pi^*_\beta; \mathcal{D}_i^{\text{eff}}) \geq f(\pi', \pi', \mathcal{D}_i^{\text{eff}}) \tag{9}$$

$$\implies \forall \pi' \in \Pi_{\text{relabel}}, \quad J_{\mathcal{D}_i^{\text{eff}}}(\pi^*) - \alpha D(\pi^*, \pi^*_\beta) \geq J_{\mathcal{D}_i^{\text{eff}}}(\pi'). \tag{10}$$

And additionally, we obtain:

$$\forall \pi' \in \Pi_{\text{relabel}}, \quad f(\pi^*, \pi^*_\beta; \mathcal{D}_i^{\text{eff}}) \geq f(\pi^*, \pi'; \mathcal{D}_i^{\text{eff}}), \tag{11}$$

$$\implies \forall \pi' \in \Pi_{\text{relabel}}, \quad D(\pi^*, \pi^*_\beta) \leq D(\pi^*, \pi'). \tag{12}$$

Utilizing 10, we obtain that:

$$J_{\mathcal{D}_i^{\text{eff}}}(\pi^*) - J_{\mathcal{D}_i^{\text{eff}}}(\pi_\beta) \geq \alpha D(\pi^*, \pi^*_\beta) + \left(J_{\mathcal{D}_i^{\text{eff}}}(\pi^*_\beta) - J_{\mathcal{D}_i^{\text{eff}}}(\pi_\beta)\right) \approx \alpha D(\pi^*, \pi^*_\beta) + \left(J(\pi^*_\beta) - J(\pi_\beta)\right), \tag{13}$$

where $\approx$ ignores sampling error terms that do not depend on distributional shift measures like D_{CQL} because π^*_β and π_β are behavior policies which generated the complete and part of the dataset, and hence these terms are dominated by and subsumed into the sampling error for π^*. Combining Equations 8 (by setting $\pi = \pi^*$) and 13, we obtain the following safe-policy improvement guarantee for π^*: $J(\pi^*) - J(\pi_\beta) \geq \zeta$, where ζ is given by:

$$\zeta = \mathcal{O}\left(\frac{1}{(1-\gamma)^2}\right)\mathbb{E}_{\mathbf{s}\sim d^{\pi^*}_{\mathcal{D}_i^{\text{eff}}}}\left[\sqrt{\frac{D_{\text{CQL}}(\pi^*, \pi^*_\beta)(\mathbf{s})+1}{|\mathcal{D}_i^{\text{eff}}(\mathbf{s})|}}\right] - \left[\alpha D(\pi^*, \pi^*_\beta) + \underbrace{J(\pi^*_\beta) - J(\pi_\beta)}_{\text{(a)}}\right],$$

which proves the desired result. □

Proposition A.1 indicates that when optimizing the behavior policy with Equation 4, we can improve upon the conventional safe-policy improvement guarantee (Equation 6) with standard single-task offline RL: not only do we improve via $D_{\text{CQL}}(\pi^*, \pi^*_\beta)$, since, $D_{\text{CQL}}(\pi^*, \pi^*_\beta) \leq D_{\text{CQL}}(\pi^*, \pi_\beta)$, which reduces sampling error, but utilizing this policy π^*_β also allows us to improve on term (a), since Equation 7 optimizes the behavior policy to be close to the learned policy π^* and maximizes the learned policy return $J_{\mathcal{D}_i^{\text{eff}}}(\pi^*)$ on the effective dataset, thus providing us with a high lower bound on $J(\pi^*_\beta)$. We formalize this insight as Lemma A.1 below:

Lemma A.1. *For sufficiently large α, $J_{\mathcal{D}_i^{\text{eff}}}(\pi^*_\beta) \geq J_{\mathcal{D}_i^{\text{eff}}}(\pi_\beta)$ and thus $(a) \geq 0$.*

Proof. To prove this, we note that using standard difference of returns of two policies, we get the following inequality: $J_{\mathcal{D}_i^{\text{eff}}}(\pi_\beta^*) \geq J_{\mathcal{D}_i^{\text{eff}}}(\pi^*) - C\frac{R_{\max}}{1-\gamma} D_{\text{TV}}(\pi^*, \pi_\beta^*)$. Moreover, from Equation 10, we obtain that: $J_{\mathcal{D}_i^{\text{eff}}}(\pi^*) - \alpha D(\pi^*, \pi_\beta^*) \geq J_{\mathcal{D}_i^{\text{eff}}}(\pi_\beta)$. So, if α is chosen such that:

$$\frac{CR_{\max}}{1-\gamma} D_{\text{TV}}(\pi^*, \pi_\beta^*) \leq \alpha D(\pi^*, \pi_\beta^*), \tag{14}$$

we find that:

$$J_{\mathcal{D}_i^{\text{eff}}}(\pi_\beta^*) \geq J_{\mathcal{D}_i^{\text{eff}}}(\pi^*) - C\frac{R_{\max}}{1-\gamma} D_{\text{TV}}(\pi^*, \pi_\beta^*) \geq J_{\mathcal{D}_i^{\text{eff}}}(\pi^*) - \alpha D(\pi^*, \pi_\beta^*) \geq J_{\mathcal{D}_i^{\text{eff}}}(\pi_\beta),$$

implying that $(a) \geq 0$. For the edge cases when either $D_{\text{TV}}(\pi^*, \pi_\beta^*) = 0$ or $D_{\text{CQL}}(\pi^*, \pi_\beta^*) = 0$, we note that $\pi^*(\mathbf{a}|\mathbf{s}) = \pi_\beta^*(\mathbf{a}|\mathbf{s})$, which trivially implies that $J_{\mathcal{D}_i^{\text{eff}}}(\pi_\beta^*) = J_{\mathcal{D}_i^{\text{eff}}}(\pi^*) \geq J_{\mathcal{D}_i^{\text{eff}}}(\pi_\beta)$, because π^* improves over π_β on the dataset. Thus, term (a) is positive for large-enough α and the bound in Proposition A.1 gains from this term additionally. □

Finally, we show that the sampling error term is controlled when utilizing Equation 4. We will show in Lemma A.2 that the sampling error in Proposition A.1 is controlled to be not much bigger than the error just due to variance, since distributional shift is bounded with Equation 4.

Lemma A.2. *If π^* and π_β^* obtained from Equation 4 satisfy, $D_{CQL}(\pi^*, \pi_\beta^*) \leq \varepsilon \ll 1$, then:*

$$(\$) := \mathbb{E}_{\mathbf{s}\sim d^{\pi^*}_{\mathcal{D}_i^{\text{eff}}}}\left[\sqrt{\frac{D_{CQL}(\pi^*, \pi_\beta^*)(\mathbf{s}) + 1}{|\mathcal{D}_i^{\text{eff}}(\mathbf{s})|}}\right] \leq (1+\varepsilon)^{\frac{1}{2}} \underbrace{\mathbb{E}_{\mathbf{s}\sim d^{\pi^*}_{\mathcal{D}_i^{\text{eff}}}}\left[\sqrt{\frac{1}{|\mathcal{D}_i^{\text{eff}}(\mathbf{s})|}}\right]}_{:=\textit{sampling error w/o distribution shift}}. \tag{15}$$

Proof. This lemma can be proved via a simple application of the Cauchy-Schwarz inequality. We can partition the first term as a sum over dot products of two vectors such that:

$$\begin{aligned}
(\$) &= \sum_{\mathbf{s}} \sqrt{d^{\pi^*}_{\mathcal{D}_i^{\text{eff}}}(\mathbf{s})(D_{\text{CQL}}(\pi^*, \pi_\beta^*)(\mathbf{s}) + 1)} \sqrt{\frac{d^{\pi^*}_{\mathcal{D}_i^{\text{eff}}}(\mathbf{s})}{|\mathcal{D}_i^{\text{eff}}(\mathbf{s})|}} \\
&\leq \sqrt{\left(\sum_{\mathbf{s}} d^{\pi^*}_{\mathcal{D}_i^{\text{eff}}}(\mathbf{s})(D_{\text{CQL}}(\pi^*, \pi_\beta^*)(\mathbf{s}) + 1)\right) \cdot \left(\sum_{\mathbf{s}} \frac{d^{\pi^*}_{\mathcal{D}_i^{\text{eff}}}(\mathbf{s})}{|\mathcal{D}_i^{\text{eff}}(\mathbf{s})|}\right)} \\
&= \sqrt{\mathbb{E}_{\mathbf{s}\sim d^{\pi^*}_{\mathcal{D}_i^{\text{eff}}}}\left[D_{\text{CQL}}(\pi^*, \pi_\beta^*)(\mathbf{s}) + 1\right] \mathbb{E}_{\mathbf{s}\sim d^{\pi^*}_{\mathcal{D}_i^{\text{eff}}}}\left[\frac{1}{|\mathcal{D}_i^{\text{eff}}(\mathbf{s})|}\right]} \leq (1+\varepsilon)^{0.5} \mathbb{E}_{\mathbf{s}\sim d^{\pi^*}_{\mathcal{D}_i^{\text{eff}}}}\left[\sqrt{\frac{1}{|\mathcal{D}_i^{\text{eff}}(\mathbf{s})|}}\right],
\end{aligned}$$

where we note that $\mathbb{E}_{\mathbf{s}\sim d^{\pi^*}_{\mathcal{D}_i^{\text{eff}}}}\left[D_{\text{CQL}}(\pi^*, \pi_\beta^*)(\mathbf{s})\right] = D_{\text{CQL}}(\pi^*, \pi_\beta^*) \leq \varepsilon$ (based on the given information in the Lemma) and that $\sqrt{\sum_i w_i \frac{1}{x_i}} \leq \sum_i w_i \frac{1}{\sqrt{x_i}}$ for $x_i, w_i > 0$ and $\sum_i w_i = 1$, via Jensen's inequality for concave functions. □

To summarize, combining Lemmas A.1 and A.2 with Proposition A.1, we conclude that utilizing Equation 4 controls the increase in sampling error due to distributional shift, and provides improvement guarantees on the learned policy beyond the behavior policy of the original dataset. We also briefly now discuss the comparison between CDS and complete data sharing. Complete data sharing would try to reduce sampling error by increasing $|\mathcal{D}_i^{\text{eff}}(\mathbf{s})|$, but then it can also increase distributional shift, $D_{\text{CQL}}(\pi^*, \pi_\beta^*)$ as discussed in Section 3. On the other hand, CDS increases the dataset size while also controlling for distributional shift (as we discussed in the analysis above), making it enjoy the benefits of complete data sharing and avoiding its pitfalls, intuitively. On the other hand, no data sharing will just incur high sampling error due to limited dataset size.

A.2. From Equation 4 to Practical CDS (Equation 5)

The goal of our practical algorithm is to convert Equation 4 to a practical algorithm while retaining the policy improvement guarantees derived in Proposition A.1. Since our algorithm does not utilize any estimator for dataset counts $|\mathcal{D}_i^{\text{eff}}(\mathbf{s})|$, and since we operate in a continuous state-action space, our goal is to retain the guarantees of increased return of π_β^*, while also avoiding sampling error.

With this goal, we first need to relax the state-distribution in Equation 4: while both $J_{\mathcal{D}_i^{\text{eff}}}(\pi)$ and $D_{\text{CQL}}(\pi, \pi_\beta)$ are computed as expectations under the marginal state-distribution of policy $\pi(\mathbf{a}|\mathbf{s})$ on the MDP defined by the dataset $\mathcal{D}_i^{\text{eff}}$, for deriving a practical method we relax the state distribution to use the dataset state-distribution $d^{\pi_\beta^*}$ and rewrite the objective in accordance with most practical implementations of actor-critic algorithms [12, 1, 26, 21, 44] below:

$$\text{(Practical Equation 4)} \quad \max_{\pi} \max_{\pi_\beta \in \Pi_{\text{relabel}}} \mathbb{E}_{\mathbf{s} \sim \mathcal{D}_i^{\text{eff}}} [\mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s}, \mathbf{a})] - \alpha D(\pi(\cdot|\mathbf{s}), \pi_\beta(\cdot|\mathbf{s}))] \tag{16}$$

This practical approximation in Equation 16 is even more justified with conservative RL algorithms when a large α is used, since a larger α implies a smaller value for $D(\pi^*, \pi_\beta^*)$ found by Equation 4, which in turn means that state-distributions $d^{\pi_\beta^*}$ and d^{π^*} are close to each other [64]. Thus, our policy improvement objective optimizes the policies π and π_β by maximizing the conservative Q-function: $\hat{Q}^\pi(\mathbf{s}, \mathbf{a}) = Q(\mathbf{s}, \mathbf{a}) - \alpha \left(\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi_\beta(\mathbf{a}|\mathbf{s})} - 1 \right)$, that appears inside the expectation in Equation 16. While optimizing the policy π with respect to this conservative Q-function $\hat{Q}^\pi(\mathbf{s}, \mathbf{a})$ is equivalent to a standard policy improvement update utilized by most actor-critic methods [21, 26, 38], we can optimize $\hat{Q}^\pi(\mathbf{s}, \mathbf{a})$ with respect to $\pi_\beta \in \Pi_{\text{relabel}}$ by relabeling only those transitions $(\mathbf{s}, \mathbf{a}, r_i', \mathbf{s}') \in \mathcal{D}_{j \to i}$ that increase the expected conservative Q-value $\mathbb{E}_{\mathbf{s} \sim \mathcal{D}_i^{\text{eff}}} \left[\mathbb{E}_{\mathbf{a} \sim \pi_\beta(\cdot|\mathbf{s})} \left[\hat{Q}^\pi(\mathbf{s}, \mathbf{a}) \right] \right]$. Note that we relaxed the expectation $\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})$ to $\mathbf{a} \sim \pi_\beta(\mathbf{a}|\mathbf{s})$ in this expectation, which can be done upto a lower-bound of the objective in Equation 16 for a large α, since the resulting policies π and π_β are close to each other.

The last step in our practical algorithm is to modify the solution of Equation 16 to still retain the benefits of reduced sampling error as discussed in Proposition A.1. To do so, we want to relabel as many points as possible, thus increasing $|\mathcal{D}_i^{\text{eff}}(\mathbf{s})|$, which leads to reduced sampling error. Since quantifying $|\mathcal{D}_i^{\text{eff}}(\mathbf{s})|$ in continuous state-action spaces will require additional machinery such as density-models, we avoid these for the sake of simplicity, and instead choose to relabel every datapoint $(\mathbf{s}, \mathbf{a}) \in \mathcal{D}_{j \to i}$ that satisfies $Q^\pi(\mathbf{s}, \mathbf{a}; i) \geq \mathbb{E}_{\mathbf{s}, \mathbf{a} \sim \mathcal{D}_i}[\hat{Q}^\pi(\mathbf{s}, \mathbf{a}; i)] \geq 0$ to task i. These datapoints definitely increase the conservative Q-value and hence increase the objective in Equation 16 (though do not globally maximize it), while also enjoying properties of reduced sampling error (Proposition A.1). This discussion motivates our practical algorithm in Equation 5.

B. Related Work

Offline RL [14, 60, 39, 42] 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 [32, 50, 59, 68, 33], NLP [29, 30], recommender systems & advertising [70, 22, 7, 76, 77] and healthcare [65, 78]. The major challenge in offline RL is the distribution shift problem [20, 36, 38], 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 [20, 48, 29, 80, 90, 36, 66, 55], variants of importance sampling based algorithms [58, 72, 47, 73, 52], uncertainty quantification on Q-values [2, 36, 80, 42], learning conservative Q-functions [38, 35], and performing model-based training with penalty on out-of-distribution states [34, 88, 51, 4, 74, 59, 41, 89]. While current benchmarks in offline RL [19, 25] 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 [79, 54, 75, 15, 27, 86, 83, 85, 33, 69] 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 [33], prior works have observed major challenges in multi-task RL, in particular, the optimization challenge [27, 63, 86]. 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 [61, 75, 23, 83], separate shared and task-specific modules with theoretical guarantees [13] and incorporating additional supervision [69].

Dataset types / Tasks	Dataset Size	Avg Return		$D_{\mathrm{KL}}(\pi, \pi_\beta)$	
		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 2. 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.

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 [33]. 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 [3, 31, 57, 62, 16, 43, 33, 8] 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 [33] or based on metrics such as high Q-values of the relabeled data [16, 43], human domain knowledge [33], and distance to states or images in goal-conditioned settings [3, 57, 53, 46, 71, 45, 28, 49, 84, 8]. 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 [8]. Our work extends the data sharing idea to multi-task offline RL settings and leverages conservatism used in offline RL methods [38] to ensure that data sharing improves multi-task learning performance and does not introduce excessive distribution shift.

C. Empirical Analysis of Data Sharing in Offline Multi-Task RL

In this section, we first describe the experimental setup and then discuss the results and possible explanations for the observed behavior, which connects to the derivation our method in Section 4.

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 [5]. 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 [33, 82, 67]. 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 2, 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 [26] 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 [19].

We train a single-task policy $\pi_{\mathrm{CQL}}(\mathbf{a}|\mathbf{s}, i)$ with CQL [38] as the base offline RL method along with two forms of data-sharing, as shown in Table 2: 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, $D(\pi(\cdot|\cdot, i), \pi_\beta^{\mathrm{eff}}(\cdot|\cdot, i))$, for $\pi = \pi_{\mathrm{CQL}}(\mathbf{a}|\mathbf{s}, i)$, 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 2. 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 2). However, a closer look at Table 2 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.

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 2. 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.

D. Experimental Evaluation

We conduct experiments to answer five main questions: (1) can CDS prevent degradation in performance when sharing data as observed in Section 3?, (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** [16], 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 [33] (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 [38] as the base offline RL algorithm for all methods. For more details on the experiment set-up and hyperparameters, see Appendix E.

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 [5] 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 [88]. To answer question (3), we also evaluate on robotic manipulation domains using environments from the Meta-World benchmark [87]. 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 [19]. The set of goals is a fixed discrete set of size 7 and 3 for large and medium mazes, respectively. Following Fu et al. [19], a reward of +1 is given and the episode terminates

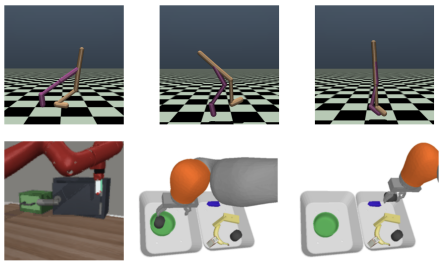

Figure 2. Our environments (from left to right): walker2d `run forward`, walker2d `run backward`, walker2d `jump`, Meta-World `door open/close` and `drawer open/close` and vision-based picking and placing tasks in [33].

Environment	Tasks / Dataset type	CDS (ours)	HIPI [16]	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 [87]	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 [19]	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 3. 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.

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 [33]. This environment, which was utilized by Kalashnikov et al. [33] as a representative and realistic simulation of a real-world robotic manipulation problem, consists of 10 image-based 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. [33], 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 [19] for AntMaze, and success rate both manipluation environments, averaged over 3 random seeds.

Multi-task datasets. Following the analysis in Section 3, 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 [32] 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 3, but leave the results of halfcheetah and ant to Appendix F. 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 $D_{\text{KL}}(\pi, \pi_\beta)$ in Table 4 where π_β is the behavior policy after we perform CDS to share data. As shown in Table 4, 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 3 and Table 2. Hence, CDS is able to mitigate distribution shift when sharing data and result in performance boost.

Environment	Dataset types / Tasks	$D_{\text{KL}}(\pi, \pi_\beta)$ **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 4. Measuring $D_{\text{KL}}(\pi, \pi_\beta)$ on the walker2d environment. **Sharing All** degrades the performance on task jump with limited expert data as discussed in Table 2. 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.

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 [33] 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 5. 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.

E. Experimental details

In this section, we provide the training details of CDS in Appendix E.1 and also include the details on the environment and datasets that we use for the evaluation in Appendix E.2. Finally, we include the discussion on the compute information in Appendix E.3. We also show the error bars of our experimental results in Table 6.

E.1. Training 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:

$$w_{\text{CDS}}(\mathbf{s}, \mathbf{a}; j \to i) := \sigma\left(\frac{\Delta(\mathbf{s}, \mathbf{a}; j \to i)}{\tau}\right), \tag{17}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ and τ is a temperature hyperparameter. This temperature parameter is chosen in a fully automatic manner utilizing adaptive temperature scaling from prior work [37] and we describe this scheme in detail in Appendix E. We also apply w_{CDS} to transitions from the same task with 0.5 probability. We use the following objectives for training the

Task Name	CDS (ours)	HIPI [16]	Skill [33]	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 5. Results for multi-task vision-based robotic manipulation domains in [33]. 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 [33] (**Skill**) and other data sharing methods on the average task success rate (highlighted in gray) and 9 out of 10 per-task success rate.

critic and the policy:

$$\hat{Q}^{k+1} \leftarrow \arg\min_{\hat{Q}} \mathbb{E}_{i\sim[N]}\Big[\beta\Big(\mathbb{E}_{j\sim[N]}\Big[\mathbb{E}_{\mathbf{s}\sim\mathcal{D}_j,\mathbf{a}\sim\mu(\cdot|\mathbf{s},i)}\Big[w_{\text{CDS}}(\mathbf{s},\mathbf{a};j\to i)\hat{Q}(\mathbf{s},\mathbf{a},i)\Big] - \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}_j}\Big[w_{\text{CDS}}(\mathbf{s},\mathbf{a};j\to i)\hat{Q}(\mathbf{s},\mathbf{a},i)\Big]\Big]\Big) + \frac{1}{2}\mathbb{E}_{j\sim[N],(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}_j}\Big[w_{\text{CDS}}(\mathbf{s},\mathbf{a};j\to i)\Big(\hat{Q}(\mathbf{s},\mathbf{a},i) - \widehat{\mathcal{B}}^{\pi}\hat{Q}^{k}(\mathbf{s},\mathbf{a},i)\Big)^2\Big]\Big],$$

and

$$\pi \leftarrow \arg\max_{\pi'} \mathbb{E}_{i\sim[N]}\Big[\mathbb{E}_{j\sim[N],\mathbf{s}\sim\mathcal{D}_j,\mathbf{a}\sim\pi'(\cdot|\mathbf{s},i)}\Big[w_{\text{CDS}}(\mathbf{s},\mathbf{a};j\to i)\hat{Q}^{\pi}(\mathbf{s},\mathbf{a},i)\Big]\Big],$$

where β is the coefficient of the CQL penalty on distribution shift, μ is a wide sampling distribution as in CQL and $\widehat{\mathcal{B}}$ is the sample-based Bellman operator.

To compute the relabeling weight $w_{\text{CDS}}(\mathbf{s},\mathbf{a};j\to i) := \sigma\left(\frac{\Delta(\mathbf{s},\mathbf{a};j\to i)}{\tau}\right)$, we need to pick the value of the temperature term τ. Instead of tuning τ manually, we follow the the adaptive temperature scaling scheme from [37]. Specifically, we compute an exponential running average of $\Delta(\mathbf{s},\mathbf{a};j\to i)$ with decay 0.995 for each task and use it as τ. We additionally clip the adaptive temperature term with a minimum and maximum threshold, which we tune manually. For multi-task halfcheetah, walker2d and ant, we clip the adaptive temperature such that it lies within $[10,\infty]$, $[5,\infty]$ and $[10,25]$ respectively. For the multi-task Meta-Wold experiment, we use $[1,50]$ for the clipping. For multi-task Antmaze, we used a range of $[10,\infty]$ for all the domains. We do not clip the temperature term on vision-based domains.

For state-based experiments, we use a stratified batch with 128 transitions for each task for the critic and policy learning. For each task i, we sample 64 transitions from $\mathcal{D}_i$ and another 64 transitions from $\cup_{j\neq i}\mathcal{D}_{j\to i}$, i.e. the relabeled datasets of all the other tasks. When computing $\Delta(\mathbf{s},\mathbf{a};j\to i)$, we only apply the weight to relabeled data on multi-task Meta-World environments and multi-task vision-based robotic manipulation tasks while also applying the weight to the original data drawn from $\mathcal{D}_i$ with 50% chance for each task $i\in[N]$ in the remaining domains.

We use CQL [38] as the base offline RL algorithm. On state-based experiments, we mostly follow the hyperparameters provided in prior work [38]. One exception is that on the multi-task ant domain, we set $\beta = 5.0$ and on the other two locomotion environments and the multi-task Meta-World domain, we use $\beta = 1.0$. On multi-task AntMaze, we use the Lagrange version of CQL, where the multiplier β is automatically tuned against a pre-specific constraint value on the CQL loss equal to $\tau = 5.0$. We use a policy learning rate $1e-4$ and a critic learning rate $3e-4$ as in [38]. On the vision-based environment, instead of using the direct CQL algorithm, we follow [8] and sample unseen actions according to the soft-max distritbution of the Q-values and set its Q target value to 0. This algorithm can be viewed the version of CQL with $\beta = 1.0$ in Eq.1 in [38], i.e. removing the term of negative expected Q-values on the dataset. We follow the other hyperparameters from prior work [32, 8, 33].

For the choice architectures, in the domains with low-dimensional state inputs, we use 3-layer feedforward neural networks with 256 hidden units for both the Q-networks and the policy. We append a one-hot task vector to the state of each environment. For the vision-based experiment, our Q-network architecture follows from multi-headed convolutional

networks used in MT-Opt [33]. For the observation input, we use images with dimension $472 \times 472 \times 3$ along with additional state features $(g_{\text{status}}, g_{\text{height}})$ as well as the one-hot task vector as in [33]. For the action input, we use Cartesian space control of the end-effector of the robot in 4D space (3D position and azimuth angle) along with two discrete actions for opening/closing the gripper and terminating the episode respectively. More details can be found in [32, 33].

E.2. Environment and dataset details

In this subsection, we discuss the details of how we set up the multi-task environment and how we collect the offline datasets. We want to acknowledge that all datasets with state inputs use the MIT License. We share the datasets of our state-based experiments here anonymously[1].

Multi-task locomotion domains. We construct the environment by changing the reward function in [5]. On the halfcheetah environment, we follow [88] and set the reward functions of task `run forward`, `run backward` and `jump` as $r(s,a) = \max\{v_x, 3\} - 0.1 * \|a\|_2^2$, $r(s,a) = -\max\{v_x, 3\} - 0.1 * \|a\|_2^2$ and $r(s,a) = -0.1 * \|a\|_2^2 + 15 * (z - \text{init z})$ respectively where v_x denotes the velocity along the x-axis and z denotes the z-position of the half-cheetah and init z denotes the initial z-position. Similarly, on walker2d, the reward functions of the three tasks are $r(s,a) = v_x - 0.001 * \|a\|_2^2$, $r(s,a) = -v_x - 0.001 * \|a\|_2^2$ and $r(s,a) = -\|v_x\| - 0.001 * \|a\|_2^2 + 10 * (z - \text{init z})$ respectively. Finally, on ant, the reward functions of the three tasks are $r(s,a) = v_x - 0.5 * \|a\|_2^2 - 0.005 *$ contact-cost, $r(s,a) = -v_x - 0.5 * \|a\|_2^2 - 0.005 *$ contact-cost and $r(s,a) = -\|v_x\| - 0.5 * \|a\|_2^2 - 0.005 *$ contact-cost $+ 10 * (z - \text{init z})$.

On each of the multi-task locomotion environment, we train each task with SAC [26] for 500 epochs. For medium-replay datasets, we take the whole replay buffer after the online SAC is trained for 100 epochs. For medium datasets, we take the online single-task SAC policy after 100 epochs and collect 500 trajectories with the medium-level policy. For expert datasets, we take the final online SAC policy and collect 5 trajectories with it for walker2d and halfcheetah and 20 trajectories for ant.

Meta-World domains. We take the `door open`, `door close`, `drawer open` and `drawer close` environments from the open-sourced Meta-World [87] repo[2]. We put both the door and the drawer on the same scene to make sure the state space of all four tasks are shared. For offline training, we use sparse rewards for each task by replacing the dense reward defined in Meta-World with the success condition defined in the public repo. Therefore, each task gets a reward of 1 if the task is fully completed and 0 otherwise.

For generating the offline datasets, we train each task with online SAC using the dense reward defined in Meta-World for 500 epochs. For medium-replay datasets, we take the whole replay buffer of the online SAC until 150 epochs. For the expert datasets, we run the final online SAC policy to collect 10 trajectories.

AntMaze domains. We take the `antmaze-medium-play` and `antmaze-large-play` datasets from D4RL [19] and convert the datasets into multi-task datasets in two ways. In the undirected version of these tasks, we split the dataset randomly into equal sized partitions, and then assign each partition to a particular randomly chosen task. Thus, the task data observed in the data for each task is largely unsuccessful for the particular task it is assigned to and effective data sharing is essential for obtaining good performance. The second setting is the directed data setting where a trajectory in the dataset is marked to belong to the task corresponding to the actual end goal of the trajectory. A sparse reward equal to +1 is provided to an agent when the current state reaches within a 0.5 radius of the task goal as was used default by Fu et al. [19].

Vision-based robotic manipulation domains. Following MT-Opt [33], we use sparse rewards for each task, i.e. reward 1 for success episodes and 0 otherwise. We define successes using the success detectors defined in [33]. To collect data for vision-based experiments, we train a policy for each task individually by running QT-Opt [32] with default hyperparameters until the task reaches 40% success rate for picking skills and 80% success rate for placing skills. We take the whole replay buffer of each task and combine all of such replay buffers to form the multi-task offline dataset with total 100K episodes where each episode has 25 transitions.

E.3. Computation Complexity

For all the state-based experiments, we train CDS on a single NVIDIA GeForce RTX 2080 Ti for one day. For the image-based robotic manipulation experiments, we train it on 16 TPUs for three days.

[1] The datasets of the state-based experiments are anonymously shared here: https://drive.google.com/file/d/17xHVcaxKF4imqP5c0R2BGWIsNCDjP4em/view?usp=sharing

[2] The Meta-World environment can be found at the public repo https://github.com/rlworkgroup/metaworld

Environment	Tasks / Dataset type	CDS (ours)	HIPI [16]	Sharing All	No Sharing
halfcheetah	run forward / medium-replay	2587.7±3.4	**2626.1**±4.2	2605.0±1.2	**2632.5**±1.2
	run backward / medium	2519.5±32.0	**2634.4**±1.2	**2636.7**±1.2	**2630.7**±35.6
	jump / expert	**4298.2**±66.6	4113.4±78.1	712.3±1874.6	-1978.3±3573.9
	average	**3135.1**±14.2	**3124.7**±27.0	1984.7±616.6	1095.0±1191.6
walker2d	run forward / medium-replay	**1100.7**±473.3	692.2±73.3	701.4±63.2	590.1±30.5
	run backward / medium	638.4±82.6	664.9±24.3	**756.7**±15.6	614.7±34.7
	jump / expert	1538.4±290.7	**1604.4**±85.5	885.1±116.0	1575.2±113.0
	average	**1092.5**±148.2	987.2±37.1	781±57.2	926.6±46.4
ant	run forward / medium-replay	2350.1±91.9	**2658.9**±63.4	1175.0±184.5	2126.7±160.3
	run backward / medium	1435.7±70.7	1208.2±64.7	1488.7±142.4	**2021.7**±14.1
	jump / expert	**2781.3**±392.6	2670.4±61.0	133.8±678.2	495.8±134.9
	average	**2189.0**±107.9	**2179.2**±9.7	932.5±179.7	1548.1±84.1
Meta-World [87]	door open / medium-replay	**57.3%**±17.2%	30.7%±43.4%	19.1%±25.2%	0%±0%
	door close / expert	**33.3%**±18.0%	0%±0%	21%±25.9%	2%±2.8%
	drawer open / expert	76%±9.3%	45.3%±32.2%	**85.5%**±7.3%	0%±0%
	drawer close / medium-replay	99.7%±0.5%	66.7%±47.1%	**100%**±0%	5.7%±4.9%
	average	**66.6%**±9.1%	35.7%±26.6%	56.4%±7.5%	1.9%±0.8%
AntMaze [19]	large maze (7 tasks) / undirected	**0.23** ± 0.17	0.01 ± 0.00	0.17 ± 0.12	0.13 ± 0.09
	large maze (7 tasks) / directed	**0.24** ± 0.05	0.12 ±0.07	**0.21** ± 0.08	**0.23** ± 0.20
	medium maze (3 tasks) / undirected	**0.37** ± 0.11	0.07 ± 0.00	0.23 ± 0.15	0.22 ± 0.12
	medium maze (3 tasks) / directed	**0.19** ± 0.08	0.08 ± 0.00	0.12 ± 0.10	**0.17** ± 0.20

Table 6. Results for multi-task locomotion, robotic manipulation and navigation environments with low-dimensional state inputs. On locomotion environments (halfcheetah, walker2d and ant), we include three tasks, run forward, run backward and jump, are provided in each environment. We use medium-replay datasets for task run forward, medium datasets for task run backward, and expert datasets with limited data for task jump. On the multi-task robotic manipulation domain, we consider four tasks from Meta-World [87], door open, door close, drawer open and drawer close with medium-replay, expert, medium-replay and expert datasets respectively. Similar to locomotion tasks, we also used limited amount of expert trajectories for the expert dataset. On the antmaze navigation task, we consider two maze layouts (medium/large) from D4RL [19], and distribute existing D4RL data either randomly across tasks (undirected) or based on relevance of the trajectory in the dataset to the task of interest (directed). Reported is the average return for locomotion tasks, normalized score as proposed in [19] for AntMaze or success rate for Meta-World environments, averaged over 3 random seeds, ± 1 standard deviation. We include both per-task performance and the overall performance averaged across tasks (highlighted in gray). We bold the highest score across all methods as well as methods with scores within 1% range of the mean of the highest score except that in the directed setting on AntMaze, we bold CDS , **No Sharing** and/or **Sharing All** due to the similar mean score and large standard deviation. CDS performs achieves the best or comparable performance on all of these multi-task environments.

F. Experimental results on additional multi-task locomotion domains

In Table 6, we provide the full results in the stated-based multi-task domains with the addition of two multi-task locomotion domains, halfcheetah and ant. CDS outperforms or achieves comparable performance on all of the low-dimensional environments.