DATA SHARING WITHOUT REWARDS IN MULTI-TASK OFFLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline reinforcement learning (RL) bears the promise to learn effective control policies from static datasets but is thus far unable to learn from large databases of heterogeneous experience. The multi-task version of offline RL enables the possibility of learning a single policy that can tackle multiple tasks and allows the algorithm to share offline data across tasks. Recent works indicate that sharing data between tasks can be highly beneficial in multi-task learning. However, these benefits come at a cost – for data to be shared between tasks, each transition must be annotated with reward labels corresponding to other tasks. This is particularly expensive and unscalable, since the manual effort in annotating reward grows quadratically with the number of tasks. Can we retain the benefits of data sharing without requiring reward relabeling for every task pair? In this paper, we show that, perhaps surprisingly, under a binary-reward assumption, simply utilizing data from other tasks with constant reward labels can not only provide substantial improvement over only using the single-task data and previously proposed success classifiers, but it can also reach comparable performance to baselines that take advantage of the oracle multi-task reward information. We also show that this performance can be further improved by selectively deciding which transitions to share, again without introducing any additional models or classifiers. We discuss how these approaches relate to each other and baseline strategies under various assumptions on the dataset. Our empirical results show that it leads to improved performance across a range of different multi-task offline RL scenarios, including robotic manipulation from visual inputs and ant-maze navigation.

1 INTRODUCTION

Offline reinforcement learning (RL) provides the promise of a fully data-driven framework for learning performant policies. To avoid costly active data collection and exploration, offline RL methods utilize a previously collected dataset to extract the best possible behavior, making it feasible to use RL to solve real-world problems where active exploration is expensive, dangerous, or otherwise infeasible (Zhan et al., 2021; de Lima & Krohling, 2021; Wang et al., 2018; Kalashnikov et al., 2018). However, this concept is only viable when a significant amount of data for the target task is available in advance. A more realistic scenario might allow for a much smaller amount of task-specific data, combined with a large amount of task-agnostic data, that is not labeled with task rewards and some of which may not be relevant. For example, if our goal is to train a robot to perform a new manipulation task (e.g., cutting an onion), we might have some data of the robot (suboptimally) attempting that task, perhaps collected under human teleoperation and manually labeled with rewards, combined with plentiful data of other tasks, some of which might be structurally related (e.g., picking up an onion, or cutting a carrot). This scenario presents several questions: How do we decide which prior data should be included when learning the new task? And how do we determine which reward labels to use for this prior data?

Prior methods have offered several potential answers to these two questions, typically in isolation. For the first question, it has been recently observed that a naïve sharing strategy of sharing data from all tasks can be highly suboptimal (Kalashnikov et al., 2021), and some works have proposed both manual (Kalashnikov et al., 2021) and automated (Yu et al., 2021a; Eysenbach et al., 2020) data-sharing strategies that prioritize the most structurally similar prior data. Most such methods assume that this shared data can be automatically relabeled with the reward function for the new task (Kalashnikov et al., 2021; Yu et al., 2021a; Eysenbach et al., 2020), but the assumption that we have access to the

functional form of this reward is a strong one: for example, in many real-world settings, the reward might require human labeling or human-provided examples (Cabi et al., 2019; Finn et al., 2016b). To this end, some prior works have proposed learning classifiers for reward labeling (Fu et al., 2018b; Xie et al., 2018; Singh et al., 2019), or other automated mechanisms (Konyushkova et al., 2020). But these mechanisms themselves add complexity and potential brittleness to the pipeline. Thus, we aim to devise a simple unified method that determines which data to share and which rewards to use, with minimal supervision and no additional modeling and learning.

In this paper, we make the potentially surprising observation that data from other tasks can be utilized with naïve constant reward labels, when the MDP consists of binary rewards. We show that this simple method, which does not involve learning any additional models or classifiers, can outperform more sophisticated techniques in practice. Our approach simply utilizes data from other tasks with a constant reward label (e.g., $r = 0$), and uses a value-aware strategy to decide which prior transitions to include for the new task. This strategy, based on the conservative data sharing (CDS) technique proposed in prior work (which assumes oracle reward access) (Yu et al., 2021a), also does not require learning any additional model and simply uses the Q-function that is already learned as part of the RL process.

Our main contribution, which we call conservative unsupervised data sharing (CUDS), is a technique for sharing data in multi-task offline RL that does not require any reward labels or reward function access for the task-agnostic data, and requires no additional model or classifier. To achieve that, our method assumes a particular form of the MDP that consists of binary rewards. We discuss the behaviors of our methods, showing that, even without ground truth reward labels, our simple data sharing scheme achieves Q-values that are lower-bounded by the Q-values obtained with sharing all data with the ground-truth rewards and can be combined CDS to selectively filter out potentially irrelevant data under different assumptions on the structure of the dataset. Our empirical evaluation conducted over various multi-task offline RL scenarios such as robotic manipulation from visual inputs and ant-maze navigation shows that this approach improves over the performance of more sophisticated techniques that either learn the reward function explicitly, or utilize other methods to propagate reward labels. In addition, we show that the proposed approach is comparable to an oracle baseline that has access to true multi-task rewards.

2 Related Work

Offline RL. Offline RL (Ernst et al., 2005; Riedmiller, 2005; Lange et al., 2012; Levine et al., 2020) considers the problem of learning a policy from a static dataset without interacting with the environment, which has shown promises in many practical applications such as robotic control (Kalashnikov et al., 2018; Mandlekar et al., 2020; Rafailov et al., 2021), NLP (Jaques et al., 2019), healthcare (Shortreed et al., 2011; Wang et al., 2018), education (de Lima & Krohling, 2021), electricity supply (Zhan et al., 2021) and UI design (Apostolopoulos et al., 2021). The main challenge of offline RL is the distributional shift between the learned policy and the behavior policy (Fujimoto et al., 2018; Kumar et al., 2019), which can cause erroneous value backups due to out-of-distribution actions generated by the learned policy. To address this issue, prior methods have constrained the learned policy to not deviate much from the behavior policy via policy regularization (Liu et al., 2020; Jaques et al., 2019; Wu et al., 2019; Zhou et al., 2020; Kumar et al., 2019; Siegel et al., 2020; Peng et al., 2019; Zhou et al., 2020; Kostrikov et al., 2021; Ghasemipour et al., 2021), conservative value functions (Kumar et al., 2020; Sinha & Garg, 2021), an auxiliary behavioral cloning loss (Fujimoto & Gu, 2021) and model-based training with conservative penalties (Yu et al., 2020d; Kidambi et al., 2020; Argenson & Dulac-Arnold, 2020; Swazinna et al., 2020; Matsushima et al., 2020; Lee et al., 2021; Yu et al., 2021b).

Multi-Task RL and data sharing. Multi-task RL (Wilson et al., 2007; Parisotto et al., 2015; Teh et al., 2017; Espeholt et al., 2018; Hessel et al., 2019; Yu et al., 2020a; Xu et al., 2020; Yang et al., 2020; Kalashnikov et al., 2021; Sodhani et al., 2021; Stooke et al., 2021) enables the goal of learning a single policy that solves multiple skills efficiently. Despite the promising results, multi-task RL suffers from three main challenges, optimization difficulties (Schaul et al., 2019; Hessel et al., 2019; Yu et al., 2020a), effective weight sharing for learning shared representations (Parisotto et al., 2015; Teh et al., 2017; Espeholt et al., 2018; Xu et al., 2020; D'Eramo et al., 2019; Sodhani et al., 2021; Stooke et al., 2021), and sharing data across different tasks (Eysenbach et al., 2020; Kalashnikov et al., 2021; Yu et al., 2021a). We consider the multi-task offline RL setting and focus

on the challenge of sharing data across different tasks. Prior works share data across tasks based on metrics such as learned Q-values (Eysenbach et al., 2020; Li et al., 2020; Yu et al., 2021a), human domain knowledge (Kalashnikov et al., 2021), the distance to the target goals in goal-conditioned settings (Andrychowicz et al., 2017; Pong et al., 2018; Nair et al., 2018; Liu et al., 2019; Sun et al., 2019; Lin et al., 2019; Huang et al., 2019; Lynch & Sermanet, 2020; Yang et al., 2021; Chebotar et al., 2021), and the learned distance with robust inference in the offline meta-RL setting (Li et al., 2019). However, all of these either require access to the functional form of the reward functions of each task in order to relabel the rewards or are limited to goal-conditioned settings. Therefore, they are not applicable to the multi-task offline RL problem that we consider where only the reward label of the originally-executed task is provided. Our work addresses this issue via simply relabeling the data shared from other tasks with a constant value and uses the conservative data sharing strategy (Yu et al., 2021a) to further improve the performance.

RL with unlabeled data. Prior works tackle the problem of learning from data without reward labels via either directly imitating expert trajectories (Pomerleau, 1988; Ross & Bagnell, 2012; Ho & Ermon, 2016), learning reward functions from expert data using inverse RL (Abbeel & Ng, 2004; Ng & Russell, 2000; Ziebart et al., 2008; Finn et al., 2016a; Fu et al., 2018a;b; Konyushkova et al., 2020), or learning a reward / value classifier that discriminates successes and failures (Xie et al., 2018; Singh et al., 2019; Eysenbach et al., 2021). These algorithms require online data collection and do not consider the offline RL setting. Singh et al. (2020) considers the single-task offline setting with both task-specific datasets and task-agnostic prior datasets and relabel the unlabeled prior data as failures since these prior transitions cannot solve the task. Our method is not limited to such single-task settings and instead considers the more general multi-task offline RL with data-sharing problem.

3 Preliminaries

Multi-task RL. Standard multi-task RL considers a multi-task Markov decision process (MDP), $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \gamma, \{R_i, i\}_{i=1}^N)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces respectively, $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ denotes the dynamics, $\gamma \in [0, 1)$ is the discount factor, and $R_1, \cdots, R_N$ correspond to reward functions of different tasks $i \in [N]$ for total number of N tasks. In our setting, we assume a binary per-task $R_i \in \{0, 1\}$, where 1 denotes success of the task and 0 otherwise. Note that the dynamics are assumed to be the same across all tasks, which is not entirely general but is indeed practical in many problem settings as noted in Yu et al. (2021a) and stands as a common assumption in prior data sharing works (Yu et al., 2021a; Kalashnikov et al., 2021; Eysenbach et al., 2020). The goal of multi-task RL is to find a task-conditioned policy $\pi(\mathbf{a}|\mathbf{s}, i)$ that expected return in a multi-task MDP: $\pi^*(\mathbf{a}|\mathbf{s}, \cdot) := \arg\max_\pi \mathbb{E}_{i\sim[N]}\mathbb{E}_{\pi(\cdot|\cdot,i)}[\sum_t \gamma^t R_i(\mathbf{s}_t, \mathbf{a}_t)]$. Note that it is possible to model the policies for each task independently as $\{\pi_1(\mathbf{a}|\mathbf{s}), \cdots, \pi_N(\mathbf{a}|\mathbf{s})\}$ without any weight sharing. In our work, we use the single task-conditioned policy to study data sharing and do not consider the weight sharing aspect, which is orthogonal to the focus of the paper, which is also noted in Yu et al. (2021a).

Multi-task offline RL and data sharing. Multi-task offline RL considers the problem of learning the multi-task policy $\pi(\mathbf{a}|\mathbf{s}, i)$ from a static multi-task dataset with $\mathcal{D} = \cup_{i=1}^N \mathcal{D}_i$ where $\mathcal{D}_\rangle = \{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j)\}_{j=1}^M$ is the per-task dataset. $\mathcal{D}_\rangle$ is generated by a behavior policy $\pi_\beta(\mathbf{a}|\mathbf{s})$, without any interaction with the environment. The most straightforward approach to learn $\pi(\mathbf{a}|\mathbf{s}, i)$ would be train it for task i only using $\mathcal{D}_i$. However, sharing data from different tasks to task i has been shown to be conducive in the multi-task offline RL setting (Kalashnikov et al., 2021; Yu et al., 2021a). To do so, prior works (Eysenbach et al., 2021; Kalashnikov et al., 2021; Yu et al., 2021a) assume access to the functional form of the reward r_i, which is a rather strong assumption that is usually impractical to specify in practical applications due to the challenge of reward specification. The next straightforward approach is to naïvely sharing data across all tasks, denoted as Sharing All. Formally, Sharing All defines the dataset of transitions relabeled from task j to task i as $\mathcal{D}_{j\to i}$ and the method can be then defined as $\mathcal{D}_i^{\text{eff}} := \mathcal{D}_i \cup (\cup_{j\neq i}\mathcal{D}_{j\to i})$, where $\mathcal{D}_i^{\text{eff}}$ denotes the effective dataset for task i. While Sharing All improves over not sharing data, as shown in Yu et al. (2021a), Sharing All leads to distributional shift that could degrade performance in certain situations (Yu et al., 2021a). In our work, we focus on the CDS (Yu et al., 2021a), which relabels data that aims to mitigate the distributional shift introduced by sharing other task data. CDS addresses such an issue by proposing a conservative data sharing strategy as follows:

$$\mathcal{D}_i^{\text{eff}} = \mathcal{D}_i \cup (\cup_{j\neq i}\{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_i) \in \mathcal{D}_{j\to i} : \Delta^\pi(\mathbf{s}, \mathbf{a}) \geq 0\}), \tag{1}$$

where $\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j$ denote the transition from $\mathcal{D}_j$, r_i denotes the reward of $\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j$ relabeled for task i and $\Delta^\pi(\mathbf{s}, \mathbf{a})$ is the condition that shares data only if the expected Q-value of the relabeled transition exceeds the top k-percentile of the Q-values of the original task data, i.e.

$$\Delta^\pi(\mathbf{s}, \mathbf{a}) := \hat{Q}^\pi(\mathbf{s}, \mathbf{a}, i) - P_{k\%}\left\{\hat{Q}^\pi(\mathbf{s}', \mathbf{a}', i) \colon \mathbf{s}', \mathbf{a}' \sim \mathcal{D}_i\right\}. \tag{2}$$

Beyond controlling the distributional shift introduced in data sharing, multi-task offline RL also needs to address the main challenge in standard offline RL, which is the distributional shift between the learned policy π and the behavior policy π_β. To handle both types of distributional shifts, CDS (Yu et al., 2021a) combines the conservative data sharing and the constrained policy optimization problem and arrives at the following objective:

$$\forall i \in [N], \quad \pi^*(\mathbf{a}|\mathbf{s}, i) := \arg\max_\pi \quad J_{\mathcal{D}_i^{\text{eff}}}(\pi) - \alpha D(\pi, \pi_\beta^{\text{eff}}), \tag{3}$$

where $\pi_\beta^{\text{eff}}(\mathbf{a}|\mathbf{s}, i)$ is the effective behavior policy for task i denoted as $\pi_\beta^{\text{eff}}(\mathbf{a}|\mathbf{s}, i) := |\mathcal{D}_i^{\text{eff}}(\mathbf{s}, \mathbf{a})|/|\mathcal{D}_i^{\text{eff}}(\mathbf{s})|$, $J_{\mathcal{D}_i^{\text{eff}}}(\pi)$ denotes the average return of policy π in the empirical MDP induced by the effective dataset, and $D(\pi, \pi_\beta^{\text{eff}})$ denotes a divergence measure (e.g., KL-divergence (Jaques et al., 2019; Wu et al., 2019), fisher divergence (Kostrikov et al., 2021), MMD distance (Kumar et al., 2019) or D_{CQL} (Kumar et al., 2020)) between the learned policy π and the effective behavior policy π_β^{eff}. While optimizing Eq. 3 with Eq. 1 as the data sharing scheme is able to mitigate distributional shift and improve over multi-task offline RL without sharing data and naïvely sharing data across all tasks as shown in Yu et al. (2021a), it requires the assumption of the access to the functional form of the reward functions, which is rather strong and make application of data sharing to real-world applications impractical. We will instead present a simple yet effective data sharing and relabeling scheme in the setting where we do not make such an assumption and instead, only have the reward labels for originally commanded task in the following section.

4 Data Sharing without Rewards in Multi-Task Offline RL

The goal of our method is to enable effective data sharing across different tasks without access to the functional form of the reward functions for each task. Data from each task is only labeled for that particular task, and we do not know a priori which data is relevant to each task. Effective data sharing therefore requires resolving two questions: (i) which data from other tasks should we use for a given task? and (ii) how do we label this data with rewards? One simple approach is to annotate all available data from other tasks with some "proxy" reward signal, and treat it no differently from data that is already labeled. That is, after relabeling with the proxy reward, we can simply put these transitions into the replay buffer of a value-based offline RL method. But how can we obtain a reliable proxy reward signal? Next, we will discuss two variants of our method in Section 4.1, understanding of both variants in Section 4.2, and practical implementations in Section 4.3.

4.1 Conservative Unsupervised Data Sharing

Prior work assumes that it is necessary to relabel prior data with some estimate of the true reward function so that the proxy reward closely reflects the true reward. We take a different approach, and instead argue that, under some assumptions, we can obtain many of the benefits of data sharing simply by labeling the multi-task data with the lowest possible reward, which we assume to be 0 without loss of generality in the binary-reward setting. We refer to this simple strategy as unsupervised data sharing (UDS). Naïve UDS prescribes sharing data from every task to every other task, and labels the shared data with a reward value of 0. Formally, for each task $i \in [N]$, we define the UDS procedure as follows:

$$\mathcal{D}_i^{\text{eff}} = \mathcal{D}_i \cup \{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, 0) \in \mathcal{D}_{j \to i} : \forall j \in [N] \setminus \{i\}\}. \tag{4}$$

Intuitively, UDS relabels data shared from other tasks with the lowest possible reward, hence making the learned Q-functions more conservative that the data sharing scheme with the oracle rewards. We will show that UDS learns Q-values that are lower-bounded by the Q-values learned by the naïve Sharing All scheme with true reward relabeling, and can be information-theoretically optimal in offline RL thanks to such conservatism. Our empirical results in Section 5 also suggests that the

benefits from data sharing outweigh the downsides of reward bias in practice. Next, we move on from the choice of proxy reward, to the choice of which data should be shared.

While UDS is simple yet effective in multi-task data sharing without reward relabing, naïvely sharing data from all other tasks with zero rewards in the offline RL setting can result in overly conservative Q-functions and policies. To further refine this strategy, we can adapt the CDS algorithm (Yu et al., 2021a) detailed in Section 3 to filter out irrelevant transitions from other tasks, and only share those transitions that are likely to be informative. We call this strategy conservative unsupervised data sharing (CUDS). We define the CUDS strategy as follows:

$$\mathcal{D}_i^{\text{eff}} = \mathcal{D}_i \cup \{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, 0) \in \mathcal{D}_{j\to i} : \Delta^\pi(\mathbf{s}, \mathbf{a}) \geq 0 \;\; \forall j \in [N] \setminus \{i\}\}. \tag{5}$$

As we will discuss in the next subsection, CUDS is able to select potentially useful transitions under certain structural assumptions on the multi-task offline dataset, and therefore produce Q-values that are not as excessively conservative as those produced by UDS. As shown in Section 5, our empirical evaluation further validates that CUDS improves over UDS and prior approaches.

Algorithm 1 (Conservative) Unsupervised Data Sharing

Require: Multi-task offline datasets $\cup_{i=1}^N \mathcal{D}_i$.
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: $\mathcal{D}_i^{\text{eff}} = \mathcal{D}_i \cup \{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, 0) \in \mathcal{D}_{j\to i} \;\; \forall j \in [N] \setminus \{i\}\}$ (UDS) or $\mathcal{D}_i^{\text{eff}} = \mathcal{D}_i \cup \{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, 0) \in \mathcal{D}_{j\to i} : \Delta^\pi(\mathbf{s}, \mathbf{a}) \geq 0 \;\; \forall j \in [N] \setminus \{i\}\}$ using Eq. 2 (CUDS).
6: Perform policy improvement by solving Eq. 3 by sampling data from $\mathcal{D}^{\text{eff}}$.

4.2 Understanding the Behavior of UDS and CUDS

In this section, we aim to understand the behavior of the UDS and CUDS. We first consider the UDS scheme, which simply shares all available data from other tasks, and labels the reward for each transition from other tasks as 0. When instantiated with CQL as the offline RL method, the Q-values of a given policy learned by UDS for each task i are the fixed point of the recursion:

$$\widehat{Q}^{k+1}(\mathbf{s}, \mathbf{a}, i) \leftarrow \widehat{r}(\mathbf{s}, \mathbf{a}, i) + \gamma \mathbb{E}_{\mathbf{s}' \sim \widehat{P}(\mathbf{s}'|\mathbf{s},\mathbf{a}), \pi(\mathbf{a}'|\mathbf{s}', i)} \left[\widehat{Q}^k(\mathbf{s}', \mathbf{a}', i)\right] - \alpha \left(\frac{\pi(\mathbf{a}|\mathbf{s}, i)}{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s}, i)} - 1\right), \tag{6}$$

where $\widehat{r}(\mathbf{s}, \mathbf{a}, i) = 0$ for all $(\mathbf{s}, \mathbf{a}) \in \mathcal{D}_{j\to i}.j \neq i$, and $\widehat{r}(\mathbf{s}, \mathbf{a}, i)$ is equivalent to the empirical reward observed otherwise (Kumar et al., 2020). We will now try to understand how UDS compares to the Sharing All strategy, where the latter can use ground truth rewards. Since, by assumption, 0 is the lowest possible reward, we would intuitively expect that UDS should be *more* conservative, compared to Sharing All that relabels with the true reward. Proposition 4.1 formalizes this insight:

Proposition 4.1. *For a policy π, let $\widehat{Q}^\pi_{UDS}(\cdot, \cdot, i)$ denote the fixed point of Eqn. 6. Then, $\widehat{Q}^\pi_{UDS}(\mathbf{s}, \mathbf{a})$ lower-bounds the Q-function $\widehat{Q}^\pi_{\text{Sharing All}}$ that would be obtained had we used the true rewards.*

We include the proof of Proposition 4.1 in Appendix A. To observe Proposition 4.1 holds intuitively, UDS underestimates the reward function at each state-action tuple when relabeling by assigning the lowest possible reward value, and hence is expected to be more conservative than Sharing All. While we may surmise that being too conservative on unlabeled data may be suboptimal, conservatism has been shown to be information-theoretically optimal (Jin et al., 2021; Rashidinejad et al., 2021) in offline RL and bandit problems. Even though an unlabeled dataset provides us with information about environment dynamics, it does not provide information about rewards, and any optimistic estimate of reward on this data may lead to poor performance in the worst case. However, as we will argue next, weighting the transitions using CUDS can further improve performance in certain cases.

To understand the behavior of CUDS, we will consider a simple abstract model of CUDS-style relabeling. In the tabular setting, this model updates the Q-function to match the target (conservative) Q-value if the transition is selected for the update, and retains the old table entry otherwise. Formally,

consider a binary vector $\mathbf{w} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$ that indicates whether a corresponding state-action pair $(\mathbf{s}, \mathbf{a})$ is utilized for the backup or not. Then, our weighted scheme performs the following backups:

$$\widehat{Q}^{k+1}(\mathbf{s}, \mathbf{a}, i) = \mathbf{w}(\mathbf{s}, \mathbf{a}) \left[\widehat{r}(\mathbf{s}, \mathbf{a}, i) - \alpha \left(\frac{\pi(\mathbf{a}|\mathbf{s}, i)}{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s}, i)} - 1 \right) + \gamma \mathbb{E}_{\mathbf{s}', \mathbf{a}' \sim \widehat{P}(\cdot|\mathbf{s},\mathbf{a}), \pi(\cdot|\mathbf{s}', i)} \left[\widehat{Q}^k(\mathbf{s}', \mathbf{a}', i) \right] \right] + (1 - \mathbf{w}(\mathbf{s}, \mathbf{a}))\widehat{Q}^k(\mathbf{s}, \mathbf{a}, i). \tag{7}$$

Equation 7 can be intuitively understood as performing a conservative backup from the actual transition observed in the dataset when the binary weight $\mathbf{w}(\mathbf{s}, \mathbf{a}) = 1$, and simply truncating the Bellman backup and retaining the previous Q-values $\widehat{Q}^k(\mathbf{s}, \mathbf{a}, i)$, otherwise. For example, CUDS performs a conservative backup with $\widehat{r}(\mathbf{s}, \mathbf{a}, i) \leq \widehat{r}_{\text{Sharing All}}(\mathbf{s}, \mathbf{a}, i)$ only on transitions where $\mathbf{w}(\mathbf{s}, \mathbf{a}) = \mathbb{I}[\Delta^\pi(\mathbf{s}, \mathbf{a}) \geq 0]$.

To understand how this affects the resulting Q-function, we consider two structural conditions on the offline dataset: **(1)** a scenario where no trajectory in the relabeled dataset for a given target task $\mathcal{D}_{j\to i}$ actually visits state-action tuples that were observed in $\mathcal{D}_i$, and **(2)** when trajectories in $\mathcal{D}_{j\to i}$ overlap with at least a fraction of state-action tuples in the original labeled data for this task $\mathcal{D}_i$. We will abstract CUDS as utilizing $\mathbf{w}^k(\mathbf{s}, \mathbf{a}) = \mathbb{I}[\widehat{Q}^k(\mathbf{s}, \mathbf{a}, i) \geq \iota]$ for some threshold ι (see Eqn. 2).

Remark 4.1 (CUDS reduces to no sharing under condition **(1)**)**.** *When the trajectories in the unlabeled, relabeled dataset do not overlap with any trajectory in the labeled dataset for a given task, any backup performed by CUDS on an unlabeled transition will eventually drive its Q-value to* 0 *as* $k \to \infty$*. Thus, CUDS weights* $\mathbf{w}^k(\mathbf{s}, \mathbf{a})$ *will eventually take on* 0 *values for such transitions, and will not be selected by the future weights, i.e.,* $\mathbf{w}^j(\mathbf{s}, \mathbf{a}) = 0 \; \forall j \geq k + 1$.

Perhaps unsurprisingly, when the unlabeled data has no overlap with the labeled data, CUDS reduces to no sharing. However, the more practically relevant case is when the unlabeled data overlaps with the labeled data. We consider the scenario when UDS has been run initially to obtain a starting set of Q-values, $\widehat{Q}^0(\mathbf{s}, \mathbf{a}, i)$, which defines the initial weight vector.

Remark 4.2 (CUDS selects more useful unlabeled transitions)**.** *When transitions* $(\mathbf{s}, \mathbf{a}, \mathbf{s}', 0) \in \mathcal{D}_{j\to i}$ *for which the next state (and the next policy action)* $(\mathbf{s}', \mathbf{a}')$ *are observed in the labeled dataset,* $\mathcal{D}_i$ *will attain large initial Q-values* $\widehat{Q}^0(\mathbf{s}, \mathbf{a}, i)$ *if executing the policy after* $(\mathbf{s}', \mathbf{a}')$ *eventually reaches the state that corresponds to a success of task* i*. Thus, CUDS will activate the condition that* $\mathbf{w}^0(\mathbf{s}, \mathbf{a}) = 1$*, and will perform conservative backups on these transitions. Two scenarios might arise:* ***(i)*** *the Q-values eventually decrease and CUDS is deactivated, i.e.,* $\exists k, \; \mathbf{w}^k(\mathbf{s}, \mathbf{a}) = 0$*, in which case this transition is discarded, or* ***(ii)*** *the learning process reaches an equilibrium where* $\mathbf{w}^k(\mathbf{s}, \mathbf{a}) = 1 \; \forall \; k$*, meaning that this relabeled transition has been incorporated into the task dataset.*

Therefore, Remark 4.2 suggests that CUDS does not have to push down Q-values at every state-action pairs and instead only performs conservative policy evaluation on state-action pairs where Q-values are high, resulting in a less conservative Q-value and policy.

4.3 Practical Implementations

We present pseudocode for UDS and CUDS in Algorithm 1. We train the Q-values with CQL to obtain conservative Q-values, and use the conservative Q-values to compute $\Delta^\pi(\mathbf{s}, \mathbf{a})$ defined in Eq. 2. For CUDS, in practice, instead of computing the hard threshold of $\Delta^\pi(\mathbf{s}, \mathbf{a}) \geq 0$ to determine data sharing, we follow Yu et al. (2021a) and transform the condition $\Delta^\pi(\mathbf{s}, \mathbf{a}) \geq 0$ into a soft weighting scheme, with weights given by $w_{\text{CUDS}}(\mathbf{s}, \mathbf{a}; j \to i) := \sigma\left(\frac{\Delta(\mathbf{s},\mathbf{a};j\to i)}{\tau}\right)$, where τ is a hyperparameter for the temperature of the sigmoid term in w_{CUDS} that is automatically selected by the running average of $\Delta(\mathbf{s}, \mathbf{a}; j \to i)$. These weights are applied to both critic and actor training. For both UDS and CUDS, we train a policy $\pi(\mathbf{a}|\mathbf{s}, i)$ where $\pi(\mathbf{a}|\mathbf{s}, i)$ could either be a single task-conditioned task with weight sharing or separate policies for each task without weight sharing. For more details of the practice implementations, see Appendix B.

5 Experiments

In this section, we present our empirical evaluation, which aims to answer the following questions: (1) Can our simple approach outperform prior methods for utilizing unlabeled offline data on multi-task

offline datasets? (2) Can the conservative data sharing strategy further improve the results achieved by our method? (3) Is our approach able to attain competitive result compared to the prior multi-task offline RL algorithm that have access to the true rewards? (4) How does CUDS compare to prior offline RL methods that directly learn representations from the multi-task offline dataset and run offline training on top of the representation?

Comparisons. We compare UDS and CUDS to a number of prior methods. We first evaluate: **No Sharing**, which performs applies standard offline RL algorithm to the multi-task setting without sharing data across tasks, **Reward Predictor**, which learns a classifier that directly predicts the reward using supervised learning, **VICE** (Fu et al., 2018c), an inverse RL method that learns a reward classifier from the labeled data and then annotates the unlabeled data with the learned classifier, and **RCE** (Eysenbach et al., 2021), a method similar to **VICE** except that **RCE** represents the Q-function as a classifier and learns the reward for unlabeled data implicitly. We adapt **VICE** and **RCE** to the multi-task offline RL setting by extracting transitions with reward labels equal to 1 and treating these datapoints as positives to learn the classifier for each task. We also train **VICE** and **RCE**, but adapt them to the offline setting using CQL, i.e. the same base offline RL method as in **UDS** and **CUDS**. Finally, to answer question (4), we conduct empirical evaluations on **ACL** (Yang & Nachum, 2021), which is a recent offline RL algorithm that performs representation learning on the offline dataset and trains the policy on top of the representation. For more details for experimental set-up, comparisons and hyperparameter settings, please see Appendix B.

5.1 Main Evaluation

To answer questions (1), (2), and (3), we perform empirical evaluations on two state-based multi-task robotic manipulation and navigation datasets and one image-based multi-task manipulation dataset introduced in prior work (Yu et al., 2021a), which we will discuss below.

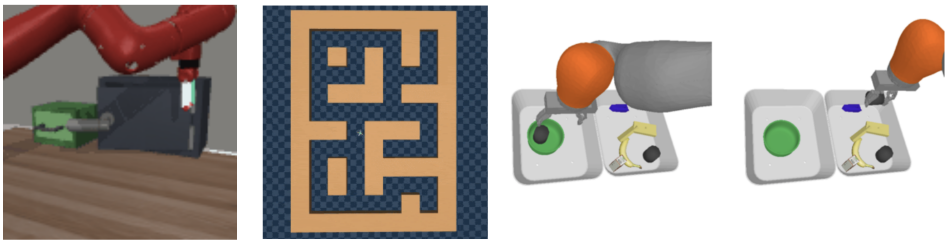

Figure 1: Environments (from left to right): Meta-World door and drawer open/close, AntMaze, and vision-based pick-place tasks.

Tasks and Datasets. Following the experimental setup in prior work (Yu et al., 2021a), we consider three domains shown in Fig. 1: (i) the Meta-World (Yu et al., 2020b) domain, which consists of four tasks of opening and closing doors and drawers; (ii) the Antmaze (Fu et al., 2020) domain, which consists of two sizes of mazes (medium and large) with 3 and 7 tasks respectively; and (iii) the multi-task visual manipulation domain, which consists of 10 tasks with different combinations of object-oriented grasping, with 7 objects (banana, bottle, sausage, milk box, food box, can and carrot), and placing the picked objects onto one of three fixtures (bowl, plate and divider plate). For all domains, we use binary rewards, where 1 denotes the successful completion of the task and 0 corresponds to failure. Note that for Meta-World, we use a fixed 200 timesteps for each episode and do not terminate the episode when receiving a reward of 1 at an intermediate timestep. In Antmaze, we terminate the episode upon seeing a reward of 1 with the maximum possible 1000 transitions per episode. We use the same datasets as prior work (Yu et al., 2021a). For Meta-World, we use large datasets with wide coverage of the state space and 152K transitions for the `door open` and `drawer close` tasks and datasets with limited (2K transitions), but optimal demonstrations for the `door close` and `drawer open` tasks. For AntMaze, following Yu et al. (2021a), we modify the datasets introduced by Fu et al. (2020) by equally dividing the large dataset into different parts for different tasks, where each task corresponds to a different goal position. For image-based manipulation, we directly use the dataset collected by Yu et al. (2021a), which contains a total of 100K RL episodes with 25 transitions for each episode, where the success rate is 40% and 80% for the picking and placing tasks, respectively. Note that the success rate of placing is higher because the robot is already holding the object at the start of the placing tasks, making the placing easier to solve.

Environment	Tasks	CUDS (ours)	UDS (ours)	VICE	RCE	No Sharing	Reward Predictor
Meta-World	door open	**61.3%**±7.9%	51.9%±25.3%	0.0%±0.0%	0.0%±0.0%	14.5%±12.7	0.0%±0.0%
	door close	54.0% ±42.5%	12.3%±27.6%	66.7%%±47.1%	0.0%±0.0%	4.0%±6.1%	**99.3%**±0.9%
	drawer open	**73.5%**±9.6%	61.8%±16.3%	0.0%±0.0%	0.0%±0.0%	16.0%±17.5%	13.3%±18.9%
	drawer close	99.3%±0.7%	**99.6%**±0.7%	19.3%±27.3%	2.7%±1.7%	99.0%±0.7%	50.3%±35.8%
	average	**71.2%** ± 11.3%	56.4%±12.8%	21.5%±0.7%	0.7%±0.4%	33.4%±8.3%	41.0%±11.9%
AntMaze	medium maze (3 tasks)	**31.5%**±3.0%	26.5%±9.1%	2.9%±1.0%	0.0%±0.0%	21.6%±7.1%	3.8%±3.8%
	large maze (7 tasks)	**18.4%**±6.1%	14.2%±3.9%	2.5%±1.1%	0.0%±0.0%	13.3% ± 8.6%	5.9%±4.1%

Table 1: Results for multi-task robotic manipulation (Meta-World) and navigation environments (AntMaze) with low-dimensional state inputs. Numbers are averaged across 6 seeds, ± the 95%-confidence interval. We take the results of **No Sharing** directly from Yu et al. (2021a). We include per-task performance for 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. Both **CUDS** and **UDS** outperforms prior vanilla multi-task offline RL approach (**No Sharing**) and reward learning methods (**Reward Predictor**, **VICE** and **RCE**)

Task Name	CUDS (ours)	UDS	No Sharing	CDS (oracle)	Sharing All (oracle)
`lift-banana`	**55.9%**±11.7%	48.6%±5.1%	20.0%±6.0%	**53.1%**±3.2%	41.8%±4.2%
`lift-bottle`	**72.9%**±12.8%	58.1%±3.6%	49.7%±8.7%	**74.0%**±6.3%	60.1%±10.2%
`lift-sausage`	**74.3%**±8.3%	66.8% ± 2.7%	60.9%±6.6%	**71.8%**±3.9%	70.0%±7.0%
`lift-milk`	73.5%±6.7%	**74.5%**±2.5%	68.4%±6.1%	**83.4%**±5.2%	72.5%±5.3%
`lift-food`	**66.3%**±8.3%	53.8%±8.8%	39.1%±7.0%	**61.4%**±9.5%	58.5%±7.0%
`lift-can`	**64.9%**±7.1%	61.0%±6.8%	49.1%±9.8%	**65.5%**±6.9%	57.7%±7.2%
`lift-carrot`	**84.1%**±3.6%	73.4%±5.8%	69.4%±7.6%	**83.8%**±3.5%	75.2%±7.6%
`place-bowl`	**83.4%**±3.6%	77.6%±1.6%	80.3%±8.6%	**81.0%**±8.1%	70.8%±7.8%
`place-plate`	**86.2%**±1.8%	78.7%±2.2%	86.1%±7.7%	**85.8%**±6.6%	78.7%±7.6%
`place-divider-plate`	**89.0%**±2.2%	80.2%±2.2%	85.0%±5.9%	**87.8%**±7.6%	79.2%±6.3%
average	**75.0%**±3.3%	67.3%±0.8%	60.8%±7.5%	**74.8%** ±6.4%	66.4%±7.2%

Table 2: Results for multi-task imaged-based robotic manipulation domains in (Yu et al., 2021a). Numbers are averaged across 3 seeds, ± the 95% confidence interval. **UDS** outperforms **No Sharing** in 7 out of 10 tasks as well as the average task performance, while performing comparably to **Sharing All**. **CUDS** further improves the performance of **UDS** and outperforms **No Sharing** in all of the 10 tasks.

Results of Question (1). The main results are in Table 1. **UDS** achieves better performance than vanilla multi-task offline RL without data sharing and compared to reward learning methods, suggesting that our simple relabeling method is effective in both multi-task manipulation and navigation domains. Since the reward learning approaches obtain similar or worse results compared to no sharing, we only compare our methods to **No Sharing** and the oracle methods in the image-based experiments. As shown in Table 2. **UDS** outperforms **No Sharing** in 7 out of 10 tasks as well as the average task performance by a significant margin. Therefore, **UDS** is able to effectively leverage unlabeled data shared from other tasks and achieves potentially surprisingly strong results compared to more sophisticated methods that handle unlabeled offline data, answering question (1).

Results of Question (2). In both the state-based and vision-based experiments shown in Table 1 and Table 2, we find that **CUDS** further improves upon the performance of **UDS**, which empirically indicates that the less conservative policy learned from CUDS's selective filtering scheme is more performant in practice. Additionally, we measure the success rates of the relabeled data in Table 5 Appendix C, measured by the oracle multi-task reward function on the Meta-World and AntMaze domain. We see that the success rates of the relabeled data are above 0% by a significant margin in most of the tasks. This suggests that UDS and CUDS are not simply relabeling with the true reward, since the relabeled data does not entirely consist of failures but rather has a significant number of successful transitions.

Results of Question (3). We also show results for oracle methods that receive true reward labels: **Sharing All**, which shares all data with ground truth rewards, and **CDS**, which uses the CDS strategy (Yu et al., 2021a) with ground truth reward relabeling. We present the results in Table 3 for state-based experiments and the last two columns on the right in Table 2 for the vision-based multi-task robotic manipulation problem. Both **CUDS** and **UDS** achieves competitive results compared to **CDS** and **Sharing All**, indicating that our simple relabeling scheme is able to remove the dependence of functional form of reward functions without much loss of performance due to lacking ground-truth reward access. This addresses question (3).

Results of Question (4). Finally, to answer question (4), on the Meta-World environment, we compare UDS and CUDS to ACL (Yang & Nachum, 2021). We use the version of ACL without inputting reward labels. ACL can be viewed as an alternative to our unlabeled sharing data scheme, which leverages unlabeled data for representation learning rather than sharing it directly. We show

Environment	Tasks	CUDS (ours)	UDS (ours)	CDS (oracle)	Sharing All (oracle)
Meta-World	door open	**61.3%**±7.9%	51.9%±25.3%	58.4%±9.3%	34.3%±17.9%
	door close	54.0% ±42.5%	12.3%±27.6%	**65.3%**±27.7%	48.3%±27.3%
	drawer open	**73.5%**±9.6%	61.8%±16.3%	57.9%±16.2%	55.1%±9.4%
	drawer close	99.3%±0.7%	99.6%±0.7%	99.0%±0.7%	98.8%±0.7%
	average	**71.2%** ± 11.3%	56.4%±12.8%	70.1%±8.1%	59.4%±5.7%
AntMaze	medium maze (3 tasks)	31.5%±3.0%	26.5%±9.1%	**36.7%**±6.2%	22.9%±3.6%
	large maze (7 tasks)	18.4%±6.1%	14.2%±3.9%	**22.8%** ± 4.5%	16.7% ± 7.0%

Table 3: Comparison between **UDS / CUDS** and the oracle data sharing strategies with access to the true reward functions for relabeling. We take the results **CDS** and **Sharing All** directly from Yu et al. (2021a). **CDS** (Yu et al., 2021a) and **Sharing All** (Kalashnikov et al., 2021). **UDS / CUDS** achieve competitive results compared to **CUDS** and **UDS**.

Environment	Tasks	CUDS (ours)	UDS (ours)	ACL
Meta-World	door open	**61.3%**±7.9%	51.9%±25.3%	2.8%±2.0%
	door close	54.0% ±42.5%	12.3%±27.6%	0.0%±0.0%
	drawer open	73.5%±9.6%	61.8%±16.3%	**83.2%**±14.2%
	drawer close	99.3%±0.7%	99.6%±0.7%	**100.0%**±0.0%
	average	**71.2%** ± 11.3%	56.4%±12.8%	46.4%±3.5%

Table 4: Comparison between **UDS / CUDS** and the **ACL** (Yang & Nachum, 2021) that performs representation learning on the unlabeled data instead of data sharing. Both **UDS** and **CUDS** outperforms **ACL** by a significant margin in the average task result, suggesting that sharing the unlabeled data is crucial in improving the multi-task offline RL performance compared to only using the data for learning the representation.

the comparison to ACL in Table 4. UDS and CUDS outperform ACL in the average task performance while ACL is only proficient on drawer-open and drawer-close, and it cannot solve door-open or door-close. This indicates that sharing the unlabeled data conservatively across all tasks is important in multi-task offline RL while pretraining representations on the whole multi-task offline dataset might have limited benefit. We note that UDS / CUDS are complementary to ACL and these approaches can be combined together to further improve performance, which we leave as future work.

6 Conclusion

In the paper, we present two new algorithms, UDS and CUDS, that handle the problem of how to share data across tasks without access to the functional form of the multi-task reward function in the multi-task offline RL setting. UDS lifts the strong assumption of having access to the reward of all tasks at each transition in previous works in multi-task data sharing via simply sharing data across all tasks and relabeling the reward of data from other tasks to the minimum reward in the MDP, which indicates failure of the task. CUDS further improves over UDS via applying a more sophisticated data sharing scheme (Yu et al., 2021a) that shares data only if the relabeled Q-values improve over the expected Q-values of the original task data. We justify that UDS obtains Q-values that are lower-bounded by the Q-values learned by data sharing with true reward labels and then discuss that under certain structures of offline datasets, CUDS can selectively apply conservative policy evaluation on only transitions with high Q-values, resulting in a less conservative algorithm. Empirically, we show that both CUDS and UDS significantly outperform vanilla multi-task offline RL without data sharing as well as more complex methods that learns the reward function either explicitly or implicitly on a range of robotic manipulation and navigation domains. CUDS also improves over UDS on all of the domains. Furthermore, CUDS and UDS achieve competitive results compared to data sharing methods with access to the oracle rewards. While our method removes the strong assumption on reward functions in data sharing for multi-task offline RL and enjoys both theoretical guarantees and good empirical results, it does have a few limitations. For example, UDS and CUDS are evaluated in MDPs with binary rewards. Exploring their effects in MDPs with continuous rewards will be an exciting future avenue.

Reproducibility Statement

We provided the code, data, and instructions needed to reproduce the main experimental results of the experiments with low-dimensional inputs in the supplementary material. We include all assumptions and derivations of all claims in Section 4.2 and Appendix A. For datasets used in our experiments, we discuss the details of processing the datasets in Appendix B.

REFERENCES

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004.

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.

Pavlos Athanasios Apostolopoulos, Zehui Wang, Hanson Wang, Chad Zhou, Kittipat Virochsiri, Norm Zhou, and Igor L Markov. Personalization for web-based services using offline reinforcement learning. *arXiv preprint arXiv:2102.05612*, 2021.

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

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

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.

Leandro M de Lima and Renato A Krohling. Discovering an aid policy to minimize student evasion using offline reinforcement learning. *arXiv preprint arXiv:2104.10258*, 2021.

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.

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

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.

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.

Benjamin Eysenbach, Sergey Levine, and Ruslan Salakhutdinov. Replacing rewards with examples: Example-based policy search via recursive classification. *arXiv preprint arXiv:2103.12656*, 2021.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pp. 49–58. PMLR, 2016a.

Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 512–519. IEEE, 2016b.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *International Conference on Learning Representations*, 2018a.

Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. Variational inverse control with events: A general framework for data-driven reward definition. *Conference on Neural Information Processing Systems*, 2018b.

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*, 2018c.

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

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

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

Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In *International Conference on Machine Learning*, pp. 3682–3691. PMLR, 2021.

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.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Conference on Neural Information Processing Systems*, 2016.

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.

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.

Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? In *International Conference on Machine Learning*, pp. 5084–5096. PMLR, 2021.

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*, pp. 651–673. PMLR, 2018.

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. *Conference on Robot Learning (CoRL)*, 2021.

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

Ksenia Konyushkova, Konrad Zolna, Yusuf Aytar, Alexander Novikov, Scott Reed, Serkan Cabi, and Nando de Freitas. Semi-supervised reward learning for offline reinforcement learning. *arXiv preprint arXiv:2012.06899*, 2020.

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

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*, pp. 11761–11771, 2019.

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

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

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.

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.

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

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

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

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

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

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

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)*, pp. 4414–4420. IEEE, 2020.

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.

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.

Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, 2000.

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

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.

Dean A Pomerleau. Alvinn: an autonomous land vehicle in a neural network. In *Proceedings of the 1st International Conference on Neural Information Processing Systems*, pp. 305–313, 1988.

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.

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.

Paria Rashidinejad, Banghua Zhu, Cong Ma, Jiantao Jiao, and Stuart Russell. Bridging offline reinforcement learning and imitation learning: A tale of pessimism. *arXiv preprint arXiv:2103.12021*, 2021.

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

Stephane Ross and Drew Bagnell. Agnostic system identification for model-based reinforcement learning. In *ICML*, 2012.

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.

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.

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.

Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering. *arXiv preprint arXiv:1904.07854*, 2019.

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.

Samarth Sinha and Animesh Garg. S4rl: Surprisingly simple self-supervision for offline reinforcement learning. *arXiv preprint arXiv:2103.06326*, 2021.

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

Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, pp. 9870–9879. PMLR, 2021.

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

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

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.

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.

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*, pp. 1015–1022, 2007.

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

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

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

Mengjiao Yang and Ofir Nachum. Representation matters: Offline pretraining for sequential decision making. *arXiv preprint arXiv:2102.05815*, 2021.

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.

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

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

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*, pp. 1094–1100. PMLR, 2020b.

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*, pp. 1094–1100. PMLR, 2020c.

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*, 2020d.

Tianhe Yu, Aviral Kumar, Yevgen Chebotar, Karol Hausman, Sergey Levine, and Chelsea Finn. Conservative data sharing for multi-task offline reinforcement learning. *arXiv preprint arXiv:2109.08128*, 2021a.

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*, 2021b.

Xianyuan Zhan, Haoran Xu, Yue Zhang, Yusen Huo, Xiangyu Zhu, Honglei Yin, and Yu Zheng. Deepthermal: Combustion optimization for thermal power generating units using offline reinforcement learning. *arXiv preprint arXiv:2102.11492*, 2021.

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

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

A PROOF OF PROPOSITION 4.1

Proposition A.1 (Proposition 4.1 restated)**.** *For a policy π, let $\widehat{Q}^{\pi}_{UDS}(\cdot,\cdot,i)$ denote the fixed point of Eqn. 6. Then, $\widehat{Q}^{\pi}_{UDS}(\mathbf{s},\mathbf{a})$ lower-bounds the Q-function $\widehat{Q}^{\pi}_{\text{Sharing All}}$ that would be obtained had we used the true rewards.*

Proof. Note that $\widehat{Q}^{\pi}_{\text{UDS}}(\mathbf{s},\mathbf{a})$ at iteration $k+1$ is defined in Eq. 6, which we restate it for convenience:

$$\widehat{Q}^{k+1}_{\text{UDS}}(\mathbf{s},\mathbf{a},i) \leftarrow \widehat{r}(\mathbf{s},\mathbf{a},i) + \gamma \mathbb{E}_{\mathbf{s}'\sim\widehat{P}(\mathbf{s}'|\mathbf{s},\mathbf{a}),\pi(\mathbf{a}'|\mathbf{s}',i)}\left[\widehat{Q}^{k}_{\text{UDS}}(\mathbf{s}',\mathbf{a}',i)\right] - \alpha\left(\frac{\pi(\mathbf{a}|\mathbf{s},i)}{\hat{\pi}_{\beta}(\mathbf{a}|\mathbf{s},i)} - 1\right), \tag{8}$$

whereas $\widehat{Q}^{\pi}_{\text{Sharing All}}(\mathbf{s},\mathbf{a})$ at iteration $k+1$ is defined as

$$\widehat{Q}^{k+1}_{\text{Sharing All}}(\mathbf{s},\mathbf{a},i) \leftarrow r(\mathbf{s},\mathbf{a},i) + \gamma \mathbb{E}_{\mathbf{s}'\sim\widehat{P}(\mathbf{s}'|\mathbf{s},\mathbf{a}),\pi(\mathbf{a}'|\mathbf{s}',i)}\left[\widehat{Q}^{k}_{\text{Sharing All}}(\mathbf{s}',\mathbf{a}',i)\right] \tag{9}$$

$$- \alpha\left(\frac{\pi(\mathbf{a}|\mathbf{s},i)}{\hat{\pi}_{\beta}(\mathbf{a}|\mathbf{s},i)} - 1\right). \tag{10}$$

Assume $\widehat{Q}^{0}_{\text{Sharing All}} = \widehat{Q}^{0}_{\text{UDS}}$, i.e. same Q-value initialization and $\widehat{Q}^{k}_{\text{Sharing All}} \geq \widehat{Q}^{k}_{\text{UDS}}$. Using such induction hypothesis and the fact that $\widehat{r}(\mathbf{s},\mathbf{a},i) \leq r(\mathbf{s},\mathbf{a},i)$ for all $\mathbf{s},\mathbf{a}$, we can conclude that $\widehat{Q}^{\pi}_{\text{UDS}}(\mathbf{s},\mathbf{a}) \leq \widehat{Q}^{\pi}_{\text{Sharing All}}(\mathbf{s},\mathbf{a})$. Therefore, UDS learns the Q-value that lower-bounds the Q-values learned data sharing all tasks with the ground-truth rewards. □

B DETAILS OF UDS AND CUDS

In this section, we include the details of training UDS and CUDS in Appendix B.1 as well as details on the environment and datasets used in our experiments in Appendix B.2. Finally, we discuss the compute information of UDS and CUDS in Appendix B.3. For additional details, please see our anonymous website: https://sites.google.com/view/uds-cuds/.

B.1 DETAILS ON THE TRAINING PROCEDURE

Our practical implementation of UDS optimizes the following objectives for the Q-functions and the policy:

$$\begin{aligned}\hat{Q}^{k+1} \leftarrow \arg\min_{\hat{Q}} \mathbb{E}_{i\sim[N]}\Big[\beta\Big(&\mathbb{E}_{j\sim[N]}\Big[\mathbb{E}_{\mathbf{s}\sim\mathcal{D}_j,\mathbf{a}\sim\mu(\cdot|\mathbf{s},i)}\left[\hat{Q}(\mathbf{s},\mathbf{a},i)\right] \\ &-\mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}_j}\left[\hat{Q}(\mathbf{s},\mathbf{a},i)\right]\Big]\Big) \\ &+\frac{1}{2}\mathbb{E}_{j\sim[N],(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}_j}\left[\left(\hat{Q}(\mathbf{s},\mathbf{a},i) - \left(r(\mathbf{s},\mathbf{a},i)\mathbb{1}_{\{j=i\}} + \gamma Q(\mathbf{s}',\mathbf{a}')\right)\right)^2\right]\Big],\end{aligned}$$

and $$\pi \leftarrow \arg\max_{\pi'} \mathbb{E}_{i\sim[N]}\left[\mathbb{E}_{j\sim[N],\mathbf{s}\sim\mathcal{D}_j,\mathbf{a}\sim\pi'(\cdot|\mathbf{s},i)}\left[\hat{Q}^{\pi}(\mathbf{s},\mathbf{a},i)\right]\right],$$

Similarly, CUDS optimizes the following objectives for training the critic and the policy with a soft weight:

$$\begin{aligned}\hat{Q}^{k+1} \leftarrow \arg\min_{\hat{Q}} \mathbb{E}_{i\sim[N]}\Big[\beta\Big(&\mathbb{E}_{j\sim[N]}\Big[\mathbb{E}_{\mathbf{s}\sim\mathcal{D}_j,\mathbf{a}\sim\mu(\cdot|\mathbf{s},i)}\left[w_{\text{CUDS}}(\mathbf{s},\mathbf{a};j\to i)\hat{Q}(\mathbf{s},\mathbf{a},i)\right] \\ &-\mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}_j}\left[w_{\text{CUDS}}(\mathbf{s},\mathbf{a};j\to i)\hat{Q}(\mathbf{s},\mathbf{a},i)\right]\Big]\Big) \\ &+\frac{1}{2}\mathbb{E}_{j\sim[N],(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}_j}\left[w_{\text{CUDS}}(\mathbf{s},\mathbf{a};j\to i)\left(\hat{Q}(\mathbf{s},\mathbf{a},i) - \left(r(\mathbf{s},\mathbf{a},i)\mathbb{1}_{\{j=i\}} + \gamma Q(\mathbf{s}',\mathbf{a}')\right)\right)^2\right]\Big],\end{aligned}$$

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

where β is the coefficient of the CQL penalty on distribution shift, μ is an action sampling distribution that covers the action bound as in CQL. We follow all the CQL hyperparameters used in Yu et al. (2021a).

To compute the weight $w_{\text{CUDS}}(\mathbf{s}, \mathbf{a}; j \to i) := \sigma\left(\frac{\Delta(\mathbf{s},\mathbf{a};j\to i)}{\tau}\right)$, we pick τ, i.e. the temperature term, using the exponential running average of $\Delta(\mathbf{s}, \mathbf{a}; j \to i)$ with decay 0.995 for each task following Yu et al. (2021a). Following Yu et al. (2021a) again, we clip the automatically chosen τ with a minimum and maximum threshold, which we directly use the values from Yu et al. (2021a). We use $[1, 50]$ and $[10, \infty]$ as the minimum and maximum threshold for the multi-task Meta-World and AntMaze domains respectively whereas the vision-based robotic manipulation domain does not require such clipping.

Following the training protocol in Yu et al. (2021a), for experiments with low-dimensional inputs, we use a stratified batch with 128 transitions for each task to train the Q-functions and the policy. We also balance the numbers of transitions sampled from the original task and the number of transitions drawn from other task data. Specifically, for each task i, we sample 64 transitions from $\mathcal{D}_i$ and the remaining 64 transitions from $\cup_{j\neq i}\mathcal{D}_{j\to i}$. In CUDS, for each task $i \in [N]$, we only apply w_{CUDS} to data shared from other tasks on multi-task Meta-World environments and multi-task vision-based robotic manipulation tasks while we also apply the relabeling weight to transitions sampled from the original task dataset $\mathcal{D}_i$ with 50% probability in the multi-task AntMaze domain.

Regarding the choices of the architectures, for state-based domains, we use 3-layer feedforward neural networks with 256 hidden units for both the Q-networks and the policy. We condition the policy on a one-hot task ID, which is appended to the input state. In domains with high-dimensional image inputs, we adopt the multi-headed convolutional neural networks used in Kalashnikov et al. (2021); Yu et al. (2021a). We use images with dimension $472 \times 472 \times 3$, extra state features $(g_{\text{robot_status}}, g_{\text{height}})$ and the one-hot task vector as the observations similar Kalashnikov et al. (2021); Yu et al. (2021a). Following the set-up in Kalashnikov et al. (2018; 2021); Yu et al. (2021a), we use Cartesian space control of the end-effector of the robot in 4D space (3D position and azimuth angle) along with two binary actions to open/close the gripper and terminate the episode respectively to represent the actions. For more details, see Kalashnikov et al. (2018; 2021).

B.2 Details on the environment and the datasets

In this subsection, we include the discussion of the details the environment and datasets used for evaluating UDS and CUDS. Note that all of our environment and offline datasets are from prior work (Yu et al., 2021a). We will nonetheless discuss the details to make our work self-contained. We acknowledge that all datasets with low-dimensional inputs are under the MIT License.

Multi-task Meta-World domains. We use the `door open`, `door close`, `drawer open` and `drawer close` environments introduced in Yu et al. (2021a) from the public Meta-World (Yu et al., 2020c) repo[1]. In this multi-task Meta-World environment, a door and a drawer are put on the same scene, which ensures that all four tasks share the same state space. The environment uses binary rewards for each task, which are adapted from the success condition defined in the Meta-World public repo. In this case, the robot gets a reward of 1 if it solves the target task and 0 otherwise.

We direct use the offline datasets constructed in Yu et al. (2021a), which are generated by training online SAC policies for each task with the dense reward defined in the Meta-World repo for 500 epochs. The medium-replay datasets use the whole replay buffer of the online SAC agent until 150 epochs while the expert datasets are collected by the final online SAC policy.

Multi-task AntMaze domains. Following Yu et al. (2021a), we use the `antmaze-medium-play` and `antmaze-large-play` datasets from D4RL (Fu et al., 2020) and partitioning the datasets into multi-task datasets in an undirected way defined in Yu et al. (2021a). Specifically, the dataset is randomly splitted into chunks with equal size, and then each chunk is assigned to a randomly chosen task. Therefore, under such a task construction scheme, the task data for each task is of low success rate for the particular task it is assigned to and it is imperative for the multi-task offline RL algorithm to leverage effective data sharing strategy to achieve good performance. In AntMaze, we also use a binary reward, which provides the agent a reward of +1 when the ant reaches a position within a 0.5 radius of the task goal, which is also the reward used default by Fu et al. (2020). The terminal of an episode is set to be true when a reward of +1 is observed.

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

Environment	Tasks	Oracle Success Rate of the Shared data
Meta-World	drawer open	47.4%
	door close	99.2%
	drawer open	0.1%
	drawer close	91.6%%
	average	59.5%
AntMaze	medium maze (3 tasks) average	4.3%
	large maze (7 tasks) average	1.6%

Table 5: Success rate of the data shared from other tasks to the target task determined by the ground-truth multi-task reward function.

Multi-task image-based robotic picking and placing domains. Following Kalashnikov et al. (2021); Yu et al. (2021a), we use sparse rewards for each task. That is, reward 1 is assigned to episodes that meet the success conditions and 0 otherwise. The success conditions are defined in (Kalashnikov et al., 2021). We directly use the dataset used in Yu et al. (2021a). Such a dataset is collected by first training a policy for each individual task using QT-Opt (Kalashnikov et al., 2018) until the success rate reaches 40% and 80% for picking tasks and placing tasks respectively and then combine the replay buffers of all tasks as the multi-task offline dataset. The dataset consists of a total number of 100K episodes with 25 transitions for each episode.

B.3 Computation Complexity

We train UDS and CUDS on a single NVIDIA GeForce RTX 2080 Ti for one day on the state-based domains. For the vision-based robotic picking and placing experiments, it takes 3 days to train it on 16 TPUs.

C Additional details on the quality of data shared from other tasks

We present the success rate of the data shared from other tasks to the target task computed by the oracle multi-task reward function in both the multi-task Meta-World and AntMaze domains in Table 5. Note that the success rate of `drawer close` and `door close` are particularly high since for other tasks, the drawer / door is initialized to be closed and therefore the success rate of other task data for these two tasks are almost 100% as defined by the success condition in the public Meta-World repo. Apart from these two particularly high success rates, the success rates of the shared data are consistently above 0% across all tasks in both domains. This fact suggests that UDS and CUDS are *not* relabeling with the ground truth reward where the relabeled data are actually all failures but rather performs the conservative bellman backups on relabeled data that is shown to be effective empirically.