# Appendices

## A Pseudocode of CDS

We present the summary of the pseudocode of CDS in Algorithm 1.

**Algorithm 1** CDS: Conservative Data Sharing

**Require:** Multi-task offline dataset $\cup_{i=1}^{N} \mathcal{D}_i$.
1: Randomly initialize policy $\pi_\theta(\mathbf{a}|\mathbf{s}, i)$.
2: **for** $k = 1, 2, 3, \cdots,$ **do**
3: Initialize $\mathcal{D}^{\text{eff}} \leftarrow \{\}$
4: **for** $i = 1, \cdots, N$ **do**
5: $\mathcal{D}_i^{\text{eff}} = \mathcal{D}_i \cup \{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_i) \in \mathcal{D}_{j \to i} : \Delta^{\pi}(\mathbf{s}, \mathbf{a}) \geq 0\}$ using Eq. 6 (CDS) or Eq. 18 (CDS(basic)).
6: Improve policy by solving eq. 2 using samples from $\mathcal{D}^{\text{eff}}$ to obtain $\pi_\theta^{k+1}$.

## B Analysis of CDS

In this section, we will analyze the key idea behind our method CDS (Section 5) and show that the abstract version of our method (Equation 5) provides better policy improvement guarantees than naïve data sharing and that the practical version of our method (Equation 6) approximates Equation 5 resulting in an effective practical algorithm.

### B.1 Analysis of the Algorithm in Equation 5

We begin with analyzing Equation 5, 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 [41, 39] 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 $\pi_\beta$, 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 $\pi$. 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 $D_{\text{CQL}}$ and use $D$ and $D_{\text{CQL}}$ interchangeably. Prior works [39] have shown that the optimal policy $\pi_i^*$ that optimizes Equation 1 attains a high probability safe-policy improvement guarantee, i.e., $J(\pi_i^*) \geq J(\pi_\beta) - \zeta_i$, where $\zeta_i$ is:

$$\zeta_i = \mathcal{O}\left( \frac{1}{(1-\gamma)^2} \right) \mathbb{E}_{\mathbf{s} \sim d^{\pi_i^*}_{\mathcal{D}_i}} \left[ \sqrt{\frac{D_{\text{CQL}}(\pi_i^*, \pi_\beta)(\mathbf{s}) + 1}{|\mathcal{D}_i(\mathbf{s})|}} \right] + \alpha D(\pi_i^*, \pi_\beta). \tag{7}$$

The first term in Equation 7 corresponds to the decrease in performance due to sampling error and this term is high when the single-task optimal policy $\pi_i^*$ visits rarely observed states in the dataset $\mathcal{D}_i$ and/or when the divergence from the behavior policy $\pi_\beta$ 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 $\pi$ in the empirical MDP induced by the transitions in the dataset $\mathcal{D}$. Further, let us assume that optimizing Equation 5 gives us the following policies:

$$\pi^*(\mathbf{a}|\mathbf{s}), \pi^*_\beta(\mathbf{a}|\mathbf{s}) := \arg \max_{\pi, \pi_\beta \in \Pi_{\text{relabel}}} \underbrace{J_{\mathcal{D}_i^{\text{eff}}}(\pi) - \alpha D(\pi, \pi_\beta)}_{:= f(\pi, \pi_\beta; \mathcal{D}_i^{\text{eff}})}, \tag{8}$$

where the optimized behavior policy $\pi^*_\beta$ is constrained to lie in a set of all policies that can be obtained via relabeling, $\Pi_{\text{relabel}}$, and the dataset, $\mathcal{D}_i^{\text{eff}}$ is sampled according to the state-action marginal

distribution of $\pi^*_\beta$, i.e., $\mathcal{D}^{\text{eff}}_i \sim d^{\pi^*_\beta}(\mathbf{s}, \mathbf{a})$. 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 $\pi_1$ and $\pi_2$, and a given dataset $\mathcal{D}$.

We now show the following result for CDS:

**Proposition B.1** (Proposition 5.1 restated)**.** *Let $\pi^*(\mathbf{a}|\mathbf{s})$ be the policy obtained by optimizing Equation 5, and let $\pi_\beta(\mathbf{a}|\mathbf{s})$ be the behavior policy for $\mathcal{D}_i$. Then, w.h.p. $\geq 1-\delta$, $\pi^*$ is a $\zeta$-safe policy improvement over $\pi_\beta$, i.e., $J(\pi^*) \geq J(\pi_\beta) - \zeta$, where $\zeta$ is given by:*

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

*where $\mathcal{D}^{\text{eff}}_i \sim d^{\pi^*_\beta}(\mathbf{s})$ and $\pi^*_\beta(\mathbf{a}|\mathbf{s})$ denotes the policy $\pi \in \Pi_{relabel}$ that maximizes Equation 5.*

*Proof.* To prove this proposition, we shall quantify the lower-bound on the improvement in the policy performance due to Equation 8 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 $\pi$, and a dataset $\mathcal{D}^{\text{eff}}_i$ with effective behavior policy $\pi_\beta(\mathbf{a}|\mathbf{s})$, the following bound holds [39]:

$$J(\pi) \geq J_{\mathcal{D}^{\text{eff}}_i}(\pi) - \mathcal{O}\left(\frac{1}{(1-\gamma)^2}\right) \mathbb{E}_{\mathbf{s}\sim d^{\pi}_{\mathcal{D}^{\text{eff}}_i}} \left[\sqrt{\frac{D_{\text{CQL}}(\pi, \pi^*_\beta)(\mathbf{s}) + 1}{|\mathcal{D}^{\text{eff}}_i(\mathbf{s})|}}\right], \tag{9}$$

where the $\mathcal{O}(\cdot)$ notation hides constants depending upon the concentration properties of the MDP [41] 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:

$$\forall \pi' \in \Pi_{\text{relabel}}, \quad f(\pi^*, \pi^*_\beta; \mathcal{D}^{\text{eff}}_i) \geq f(\pi', \pi', \mathcal{D}^{\text{eff}}_i) \tag{10}$$

$$\implies \forall \pi' \in \Pi_{\text{relabel}}, \quad J_{\mathcal{D}^{\text{eff}}_i}(\pi^*) - \alpha D(\pi^*, \pi^*_\beta) \geq J_{\mathcal{D}^{\text{eff}}_i}(\pi'). \tag{11}$$

And additionally, we obtain:

$$\forall \pi' \in \Pi_{\text{relabel}}, \quad f(\pi^*, \pi^*_\beta; \mathcal{D}^{\text{eff}}_i) \geq f(\pi^*, \pi'; \mathcal{D}^{\text{eff}}_i), \tag{12}$$

$$\implies \forall \pi' \in \Pi_{\text{relabel}}, \quad D(\pi^*, \pi^*_\beta) \leq D(\pi^*, \pi'). \tag{13}$$

Utilizing 11, we obtain that:

$$J_{\mathcal{D}^{\text{eff}}_i}(\pi^*) - J_{\mathcal{D}^{\text{eff}}_i}(\pi_\beta) \geq \alpha D(\pi^*, \pi^*_\beta) + \left(J_{\mathcal{D}^{\text{eff}}_i}(\pi^*_\beta) - J_{\mathcal{D}^{\text{eff}}_i}(\pi_\beta)\right) \approx \alpha D(\pi^*, \pi^*_\beta) + \left(J(\pi^*_\beta) - J(\pi_\beta)\right), \tag{14}$$

where $\approx$ ignores sampling error terms that do not depend on distributional shift measures like $D_{\text{CQL}}$ because $\pi^*_\beta$ and $\pi_\beta$ 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 $\pi^*$. Combining Equations 9 (by setting $\pi = \pi^*$) and 14, we obtain the following safe-policy improvement guarantee for $\pi^*$: $J(\pi^*) - J(\pi_\beta) \geq \zeta$, where $\zeta$ is given by:

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

which proves the desired result. □

Proposition B.1 indicates that when optimizing the behavior policy with Equation 5, we can improve upon the conventional safe-policy improvement guarantee (Equation 7) 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 $\pi^*_\beta$ also allows us to improve on term $(a)$, since Equation 8 optimizes the behavior policy to be close to the learned policy $\pi^*$ and maximizes the learned policy return $J_{\mathcal{D}^{\text{eff}}_i}(\pi^*)$ on the effective dataset, thus providing us with a high lower bound on $J(\pi^*_\beta)$. We formalize this insight as Lemma B.1 below:

**Lemma B.1.** *For sufficiently large $\alpha$, $J_{\mathcal{D}_i^{\text{eff}}}(\pi^*_\beta) \geq J_{\mathcal{D}_i^{\text{eff}}}(\pi_\beta)$ 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 11, 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 $\alpha$ is chosen such that:

$$\frac{CR_{\max}}{1-\gamma} D_{\text{TV}}(\pi^*, \pi^*_\beta) \leq \alpha D(\pi^*, \pi^*_\beta), \tag{15}$$

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 $\pi^*$ improves over $\pi_\beta$ on the dataset. Thus, term $(a)$ is positive for large-enough $\alpha$ and the bound in Proposition B.1 gains from this term additionally. □

Finally, we show that the sampling error term is controlled when utilizing Equation 5. We will show in Lemma B.2 that the sampling error in Proposition B.1 is controlled to be not much bigger than the error just due to variance, since distributional shift is bounded with Equation 5.

**Lemma B.2.** *If $\pi^*$ and $\pi^*_\beta$ obtained from Equation 5 satisfy, $D_{CQL}(\pi^*, \pi^*_\beta) \leq \varepsilon \ll 1$, then:*

$$(\$) := \mathbb{E}_{\mathbf{s}\sim d^{\pi^*}_{\mathcal{D}_i^{\text{eff}}}}\left[\sqrt{\frac{D_{CQL}(\pi^*, \pi^*_\beta)(\mathbf{s}) + 1}{|\mathcal{D}_i^{\text{eff}}(\mathbf{s})|}}\right] \leq (1+\varepsilon)^{\frac{1}{2}} \underbrace{\mathbb{E}_{\mathbf{s}\sim d^{\pi^*}_{\mathcal{D}_i^{\text{eff}}}}\left[\sqrt{\frac{1}{|\mathcal{D}_i^{\text{eff}}(\mathbf{s})|}}\right]}_{:=\textit{sampling error w/o distribution shift}}. \tag{16}$$

*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 B.1 and B.2 with Proposition B.1, we conclude that utilizing Equation 5 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 4. 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.

### B.2 From Equation 5 to Practical CDS (Equation 6)

The goal of our practical algorithm is to convert Equation 5 to a practical algorithm while retaining the policy improvement guarantees derived in Proposition B.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 $\pi^*_\beta$, while also avoiding sampling error.

With this goal, we first need to relax the state-distribution in Equation 5: 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, 46] below:

$$\text{(Practical Equation 5)} \quad \max_{\pi} \max_{\pi_\beta \in \Pi_{\text{relabel}}} \mathbb{E}_{\mathbf{s}\sim\mathcal{D}_i^{\text{eff}}}[\mathbb{E}_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] - \alpha D(\pi(\cdot|\mathbf{s}), \pi_\beta(\cdot|\mathbf{s}))] \quad (17)$$

This practical approximation in Equation 17 is even more justified with conservative RL algorithms when a large $\alpha$ is used, since a larger $\alpha$ implies a smaller value for $D(\pi^*, \pi^*_\beta)$ found by Equation 5, which in turn means that state-distributions $d^{\pi^*_\beta}$ and $d^{\pi^*}$ are close to each other [65]. Thus, our policy improvement objective optimizes the policies $\pi$ and $\pi_\beta$ 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 17. While optimizing the policy $\pi$ 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, 39], 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 17 for a large $\alpha$, since the resulting policies $\pi$ and $\pi_\beta$ are close to each other.

The last step in our practical algorithm is to modify the solution of Equation 17 to still retain the benefits of reduced sampling error as discussed in Proposition B.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 17 (though do not globally maximize it), while also enjoying properties of reduced sampling error (Proposition B.1). This discussion motivates our practical algorithm in Equation 6.

## C Experimental details

In this section, we provide the training details of CDS in Appendix C.1 and also include the details on the environment and datasets that we use for the evaluation in Appendix C.2. Finally, we include the discussion on the compute information in Appendix C.3. We also compare CDS to an offline RL with pretrained representations from multi-task datasets method [86].

### C.1 Training details

The pseudocode of CDS is summarized in Algorithm 1 in Appendix A. The complete variant of CDS can be directly implemented using the rule in Equation 6 with conservative Q-value estimates obtained via any offline RL method that constrains the learned policy to the behavior policy. For implementing CDS (basic), we reparameterize the divergence $D$ in Equation 4 to use the learned conservative Q-values. This is especially useful for our implementation since we utilize CQL as the base offline RL method, and hence we do not have access to an explicit divergence. In this case, $\Delta^\pi(\mathbf{s},\mathbf{a})$ can be redefined as, $\Delta^\pi(\mathbf{s},\mathbf{a}) :=$

$$\mathbb{E}_{\mathbf{s}'\sim\mathcal{D}^i}\left[\mathbb{E}_{\mathbf{a}'\sim\pi}[\hat{Q}(\mathbf{s}',\mathbf{a}',i)] - \mathbb{E}_{\mathbf{a}''\sim\mathcal{D}_i}[\hat{Q}(\mathbf{s}',\mathbf{a}'',i)]\right] - \left(\mathbb{E}_{\mathbf{a}'\sim\pi}[\hat{Q}(\mathbf{s},\mathbf{a}',i)] - Q(\mathbf{s},\mathbf{a},i)\right), \quad (18)$$

Equation 18 can be viewed as the difference between the CQL [39] regularization term on a given $(\mathbf{s}, \mathbf{a})$ and the original dataset for task $i$, $\mathcal{D}_i$. This CQL regularization term is equal to the divergence between the learned policy $\pi(\cdot|\mathbf{s})$ and the behavior policy $\pi_\beta(\cdot|\mathbf{s})$, therefore Equation 18 practically computes Equation 4.

Note that both variants of CDS train 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, using the resulting relabeled dataset, $\mathcal{D}_i^{\text{eff}}$. Next, we discuss the training details of the complete version of CDS.

Our practical implementation of CDS optimizes the following objectives for training the 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 $\beta$ is the coefficient of the CQL penalty on distribution shift, $\mu$ 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 $\tau$. Instead of tuning $\tau$ manually, we follow the the adaptive temperature scaling scheme from [38]. 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 $\tau$. 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 [39] as the base offline RL algorithm. On state-based experiments, we mostly follow the hyperparameters provided in prior work [39]. 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 $\beta$ 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 [39]. 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 [39], 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].

### C.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.

**Multi-task locomotion domains.** We construct the environment by changing the reward function in [5]. On the halfcheetah environment, we follow [91] 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 $r(s,a) = v_x - 0.5 * \|a\|_2^2 - 0.005 * \text{contact-cost}$, $r(s,a) = -v_x - 0.5 * \|a\|_2^2 - 0.005 * \text{contact-cost}$ and $r(s,a) = -\|v_x\| - 0.5 * \|a\|_2^2 - 0.005 * \text{contact-cost} + 10 * (z - \text{init z})$.

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 [90] repo[1]. 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.

### C.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 Meta-World environment can be found at the public repo https://github.com/rlworkgroup/metaworld

# D Visualizations, Comparisons and Additional Experiments

In this section, we perform diagnostic and ablation experiments to: **(1)** understand the efficacy of CDS when applied with other base offline RL algorithms, such as BRAC [82], **(2)** visualize the weights learned by CDS to understand if the weighting scheme induced by CDS corresponds to what we would intuitively expect on different tasks, and **(3)** compare CDS to a prior approach that performs representation learning from offline multi-task datasets and then runs vanilla multi-task RL algorithm on top of the learned representations. We discuss these experiments next.

## D.1 Applying CDS with BRAC [82], A Policy-Constraint Offline RL Algorithm

We implemented CDS on top of BRAC which is different from CQL that penalizes Q-functions. BRAC computes the divergence $D(\pi, \pi_\beta)$ in Equation 1 explicitly and penalizes the reward function $r(\mathbf{s}, \mathbf{a})$ with this value in the Bellman backups. To apply CDS to BRAC, we need to compute a conservative estimate of the Q-value as discussed in Section 5.2. While the Q-function from CQL directly provides us with this conservative estimate, BRAC does not directly learn a conservative Q-function estimator. Therefore, for BRAC, we compute this conservative estimate by explicitly subtracting KL divergence between the learned policy $\pi(a|s)$ and the behavior policy $\pi^\beta$ on state-action tuples $(s, a)$ from the learned Q-function's prediction. Formally, this means that we utilize $\hat{Q}(s, a) := Q(s, a) - \alpha \mathrm{D}_{\mathrm{KL}}(\pi(a|s), \pi^\beta(a|s))$ as our conservative Q-value estimate for BRAC. Given these conservative Q-value estimate, CDS weights can be computed directly using Equation 6.

| Environment | Tasks / Dataset type | BRAC + CDS (ours) | BRAC + No Sharing | BRAC + Sharing All |
|---|---|---|---|---|
| Meta-World [90] | door open / expert | **44.0%**±3.0% | 35.0%±25.9% | 38.0%±2.2% |
| | door close / medium-replay | **32.5%** ± 5.0% | 5.0% ± 8.6% | 8.6% ± 3.4% |
| | drawer open / medium-replay | **28.5%**±3.5% | 21.8%±5.6% | 0.0% ± 0.0% |
| | drawer close / expert | 100%±0.0% | 100.0%±0.0% | 99.0%±0.7% |
| | **average** | **52.5%**±7.4% | 22.5%±13.3% | 40.0%±5.0% |

Table 5: Applying CDS on top of BRAC. Note that CDS + BRAC imrpoves over both BRAC + Sharing All and BRAC + No sharing, indicating that CDS is effective over other offline RL algorithms such as BRAC as well. The ± values indicate the value of the standard deviation of runs, and results are averaged over three seeds.

We evaluated BRAC + CDS on the Meta-World tasks and compared it to BRAC + Sharing All and BRAC + No Sharing. We present the results in Table 5. We use ± to denote the 95%-confidence interval. As observed below, BRAC + CDS significantly outperforms BRAC with Sharing All and BRAC with No sharing. This indicates that CDS is effective on top of BRAC.

## D.2 Analyzing CDS weights for Different Scenarios

Next, to understand if the weights assigned by CDS align with our expectation for which transitions should be shared between tasks, we perform diagnostic analysis studies on the weights learned by CDS on the Meta-World and Antmaze domains.

**On the Meta-World environment**, we would expect that for a given target task, say Drawer Close, transitions from a task that involves a different object (door) and a different skill (open) would not be as useful for learning. To understand if CDS weights reflect this expectation, we compare the average CDS weights on transitions from all the other tasks to two target tasks, Door Open and Drawer Close, respectively and present the results in Table 6. We sort the CDS weights in the descending order. As shown, indeed CDS assigns higher weights to more related tasks and thus shares data from those tasks. In particular, the CDS weights for relabeling data from the task that handles the same object as the target task are much higher than the weights for tasks that consider a different object.

For example, when relabeling to the target task Door Open, datapoints from task Door Close are assigned with much higher weights than those from either task Drawer Open or task Drawer Close. This suggests that CDS filters the irrelevant transitions for learning a given task.

**On the AntMaze-large environment**, with undirected data, we visualize the CDS weight for the various tasks (goals) in the form of a heatmap and present the results in Figure 4. To generate this plot, we sample a set of state-action pairs from the entire dataset for all tasks, and then plot the weights assigned by CDS as the color of the point marker at the $(x, y)$ locations of these state-action pairs in the maze. Each plot computes the CDS weight corresponding to the target task (goal) indicated by the

| Relabeling Direction | CDS weight |
|---|---|
| door close → door open | 0.46 |
| drawer open → door open | 0.10 |
| drawer close → door open | 0.02 |
| drawer open → drawer close | 0.35 |
| door open → drawer close | 0.26 |
| door close → drawer close | 0.22 |

Table 6: On the Meta-World domain, we visualize the CDS weights of data relabeled from other tasks to the two target tasks door open and drawer close shown in the second row and third row respectively. We sort the CDS weights for relabeled tasks to a particular target task in the descending order. As shown in the table, CDS upweights tasks that are more related to the target task, e.g. manipulating the same object.

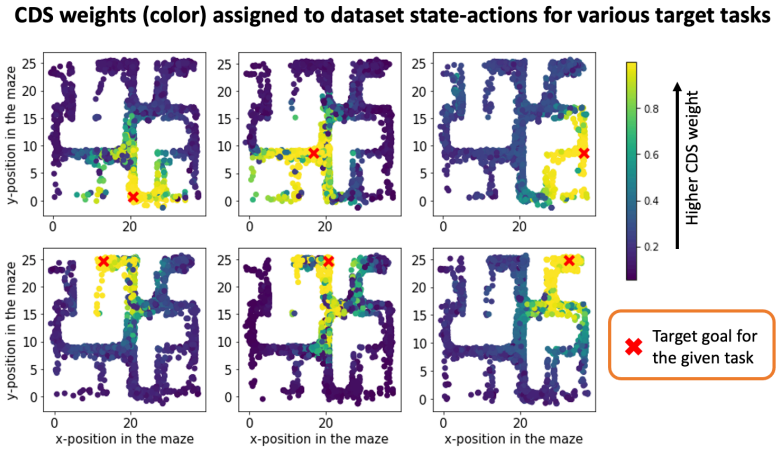


Figure 4: A visualization of the weights assigned by CDS to various transitions in the antmaze dataset for six target goals (indicated by $\times$ clustered by their spatial location. Note that CDS up-weights transitions spatially close to the target goal (indicated in the brighter yellow color), matching our expectation.

red $\times$ in the plot. As can be seen in Figure 4, CDS assigns higher weights to transitions from nearby goals as compared to transitions from farther away goals. This matches our expectation: transitions from nearby $(x, y)$ locations are likely to be the most useful in learning a particular target task and CDS chooses to share these transitions to the target task.

### D.3 Comparison of CDS with Other Alternatives to Data Sharing: Utilizing Multi-Task Datasets for Learning Pre-Trained Representations

Finally, we aim to empirically verify how other alternatives to data sharing perform on multi-task offline RL problems. One simple approach to utilize data from other tasks is to use this data to learn low-dimensional representations that capture meaningful information about the environment initially in a pre-training phase and then utilize these representations for improved multi-task RL without any specialized data sharing schemes. To assess the efficacy of this alternate approach of using multi-task offline data, in Table 7, we performed an experiment on the Meta-World domain that first utilizes the data from all the tasks to learn a shared representation using the best method, ACL [86] and then runs standard offline multi-task RL on top of this representation. We denote the method as **Offline Pretraining**. We include the average task success rates of all tasks in the table below. While the representation learning approach improves over standard multi-task RL without representation learning (**No Sharing**) consistent with the findings in [86], we still find that CDS with no representation learning outperforms this representation learning approach by a large margin on multi-task performance, which suggests that conservative data sharing is more important than pure pretrained representation from multi-task datasets in the offline multi-task setting. We finally remark that in principle, we could also utilize representation learning approaches in conjunction with data sharing strategies and systematically characterizing this class of hybrid approaches is a topic of future work.

| Environment | Tasks / Dataset type | CDS (ours) | No Sharing | Offline Petraining [86] |
|---|---|---|---|---|
| Meta-World [90] | door open / expert | **58.4%**±9.3% | 14.5%±12.7% | 48.0%±40.0% |
| | door close / medium-replay | **65.3%**±27.7% | 4.0%±6.1% | 9.5%±8.4% |
| | drawer open / medium-replay | **57.9%**±16.2% | 16.0%±17.5% | 1.3% ± 1.4% |
| | drawer close / expert | 98.8%±0.7% | 99.0%±0.7% | 96.0%±0.9% |
| | **average** | **70.1%**±8.1% | 33.4%±8.3% | 38.7%±11.1% |

Table 7: Comparison between CDS and Offline Pretraining [86] that pretrains the representation from the multi-task offline data and then runs multi-task offline RL on top of the learned representation on the Meta-World domain. Numbers are averaged across 6 seeds, ± the 95%-confidence interval. CDS significantly outperforms Offline Pretraining.