Semi-Supervised Offline Reinforcement Learning with Action-Free Trajectories

Qinqing Zheng \(^1\)  Mikael Henaff \(^1\)  Brandon Amos \(^1\)  Aditya Grover \(^2\)

Abstract

Natural agents can effectively learn from multiple data sources that differ in size, quality, and types of measurements. We study this heterogeneity in the context of offline reinforcement learning (RL) by introducing a new, practically motivated semi-supervised setting. Here, an agent has access to two sets of trajectories: labelled trajectories containing state, action and reward triplets at every timestep, along with unlabelled trajectories that contain only state and reward information. For this setting, we develop and study a simple meta-algorithmic pipeline that learns an inverse dynamics model on the labelled data to obtain proxy-labels for the unlabelled data, followed by the use of any offline RL algorithm on the true and proxy-labelled trajectories. Empirically, we find this simple pipeline to be highly successful — on several D4RL benchmarks (Fu et al., 2020), certain offline RL algorithms can match the performance of variants trained on a fully labelled dataset even when we label only 10% of trajectories which are highly suboptimal. To strengthen our understanding, we perform a large-scale controlled empirical study investigating the interplay of data-centric properties of the labelled and unlabelled datasets, with algorithmic design choices (e.g., choice of inverse dynamics, offline RL algorithm) to identify general trends and best practices for training RL agents on semi-supervised offline datasets.

1 Introduction

One of the key challenges with deploying reinforcement learning (RL) agents is their prohibitive sample complexity for real-world applications. Offline reinforcement learning (RL) can significantly reduce the sample complexity by exploiting logged demonstrations from auxiliary data sources (Levine et al., 2020). Standard offline RL assumes fully logged datasets: the trajectories are complete sequences of observations, actions, and rewards. However, contrary to curated benchmarks in use today, the nature of offline demonstrations in the real world can be highly varied. For example, the demonstrations could be misaligned due to frequency mismatch (Burns et al., 2022), use different sensors, actuators, or dynamics (Reed et al., 2022; Lee et al., 2022), or lack partial state (Ghosh et al., 2022; Rafailov et al., 2021; Mazoure et al., 2021) or reward information (Yu et al., 2022). Successful offline RL in the real world requires embracing these heterogeneous aspects for maximal data efficiency, similar to learning in humans.

In this work, we propose a new and practically motivated semi-supervised setup for offline RL: the offline dataset consists of some action-free trajectories (which we call unlabelled) in addition to the standard action-complete trajectories (which we call labelled). In particular, we are mainly interested in the case where a significant majority of the trajectories in the offline dataset are unlabelled, and the unlabelled data might have different qualities than the labelled ones. One motivating example for this setup is learning from videos (Schmeckpeper et al., 2020a;b) or third-person demonstrations (Stadie et al., 2017; Sharma et al., 2019). There are tremendous amounts of internet videos that can be potentially used to train RL agents, yet they are without action labels and are of varying quality. Notably, our setup has two key properties that differentiate it from traditional semi-supervised learning:

- First, we do not assume that the distribution of the labelled and unlabelled trajectories are necessarily identical. In realistic scenarios, we expect these to be different with unlabelled data having higher returns than labelled data e.g., videos of a human professional are easy to obtain whereas precisely measuring their actions is challenging. We replicate such varied data quality setups in some of our experiments; Figure 1.1 shows an illustration of the difference in returns between the labelled and unlabelled dataset splits using the hopper-medium-expert D4RL dataset.
- Second, our end goal goes beyond labelling the actions in the unlabelled trajectories, but rather we intend to use the unlabelled data for learning a downstream policy that is better than the behavioral policies used for generating the offline datasets.

---

\(^1\)Meta AI Research  \(^2\)UCLA. Correspondence to: Qinqing Zheng <zhengqinqing@gmail.com>.

The goal of offline RL is to learn effective policies from fixed datasets which are generated by unknown behavior policies. There are two main categories of model-free offline RL methods: value-based methods and behavior cloning (BC) based methods.

Value-based methods attempt to learn value functions based on temporal difference (TD) updates. There is a line of work that aims to port existing off-policy value-based online RL methods to the offline setting, with various types of additional regularization components that encourage the learned policy to stay close to the behavior policy. Several representative techniques include specifically tailored policy parameterizations (Fujimoto et al., 2019; Ghasemipour et al., 2021), divergence-based regularization on the learned policy (Wu et al., 2019; Jaques et al., 2019; Kumar et al., 2019), and regularized value function estimation (Nachum et al., 2019; Kumar et al., 2020; Kostrikov et al., 2021a; Fujimoto & Gu, 2021; Kostrikov et al., 2021b).

A growing body of recent work formulates offline RL as a supervised learning problem (Chen et al., 2021; Janner et al., 2021; Emmons et al., 2021). Compared with value-based methods, these supervised methods enjoy several appealing properties including algorithmic simplicity and training stability. Generally speaking, these approaches can be viewed as conditional behavior cloning methods (Bain & Sammut, 1995), where the conditioning is based on goals or returns. Similar to value-based methods, these can be extended to the online setup as well (Zheng et al., 2022) and demonstrate excellent performance in hybrid setups involving both offline data and online interactions.

**Semi-Supervised Learning**

Semi-supervised learning (SSL) is a sub-area of machine learning that studies approaches to train predictors from a small amount of labelled data combined with a large amount of unlabelled data. In
supervised learning, predictors only learn from labelled data. However, labelled training examples often require human annotation efforts and are thus hard to obtain, whereas unlabelled data can be comparatively easy to collect. The research on semi-supervised learning spans several decades. One of the oldest SSL techniques, self-training, was originally proposed in the 1960s (Fralick, 1967). There, the predictor is first trained on the labelled data. Then, at each training round, according to certain selection criteria such as model uncertainty, a portion of the unlabelled data is annotated by the predictor and added into the training set for the next round. Such process is repeated multiple times. We refer the readers to Zhu (2005); Chapelle et al. (2006); Stadie et al., 2017; Sharma et al., 2019) for comprehensive literature surveys.

Imitation Learning from Observations: There have been several works in imitation learning (IL) which do not assume access to the full set of actions, such as BCO (Torabi et al., 2018a), MoBILE (Kidambi et al., 2021), GAIFO (Torabi et al., 2018b) or third-person IL approaches (Stadie et al., 2017; Sharma et al., 2019). The recent work of Baker et al. (2022) also considered a setup where a small number of labelled actions are available in addition to a large unlabelled dataset. A key difference with our work is that the IL setup typically assumes that all trajectories are generated by an expert, unlike our offline setup. Further, some of these methods even permit reward-free interactions with the environment which is not possible in the offline setup.

Learning from Videos: Several works consider training agents with human video demonstrations (Schmeckpeper et al., 2020a;b), which are without action annotations. Distinct from our setup, some of these works allow for online interactions, assume expert videos, and more broadly, video data typically specifies agents with different embodiments.

3 Semi-Supervised Offline RL

Preliminaries: We model our environment as a Markov decision process (MDP) (Bellman, 1957) denoted by \((\mathcal{S}, \mathcal{A}, P, R, \gamma)\), where \(\mathcal{S}\) is the state space, \(\mathcal{A}\) is the action space, \(p(s_1)\) is the distribution of the initial state, \(P(s_{t+1}|s_t, a_t)\) is the transition probability distribution, \(R(s_t, a_t)\) is the deterministic reward function, and \(\gamma\) is the discount factor. At each timestep \(t\), the agent observes a state \(s_t \in \mathcal{S}\) and executes an action \(a_t \in \mathcal{A}\). The environment then moves the agent to the next state \(s_{t+1} \sim P(\cdot|s_t, a_t)\), and also returns the agent a reward \(r_t = R(s_t, a_t)\).

3.1 Proposed Setup

We assume the agent has access to a static offline dataset \(\mathcal{T}_{\text{offline}}\). The dataset consists of trajectories collected by unknown policies, which are generally suboptimal. Let \(\tau\) denote a trajectory and \(|\tau|\) denote its length. We assume that all the trajectories in \(\mathcal{T}_{\text{offline}}\) contain complete rewards and states. However, only a small subset of them contain actions.

We are interested in learning a policy by leveraging the offline dataset without interacting with the environment. This setup is analogous to semi-supervised learning, where actions serve the role of labels. Hence, we also refer to the complete trajectories as labelled data (denoted by \(\mathcal{T}_{\text{labelled}}\)) and the action-free trajectories as unlabelled data (denoted by \(\mathcal{T}_{\text{unlabelled}}\)). Further, we assume the labelled and unlabelled data are sampled from two distributions \(\mathcal{P}_{\text{labelled}}\) and \(\mathcal{P}_{\text{unlabelled}}\), respectively. In general, the two distributions can be different. One case we are particularly interested in is when \(\mathcal{P}_{\text{labelled}}\) generates low-to-moderate quality trajectories, whereas \(\mathcal{P}_{\text{unlabelled}}\) generates trajectories of diverse qualities including ones with high returns, as shown in Fig 1.1.

Our setup shares some similarities with state-only imitation learning (Ijspeert et al., 2002; Bentivegna et al., 2002; Torabi et al., 2019) in the use of action-unlabelled trajectories. However, there are two fundamental differences. First, in state-only IL, the unlabelled demonstrations are from the same distribution as the labelled demonstrations, and both are generated by a near-optimal expert policy. In our setting, \(\mathcal{P}_{\text{labelled}}\) and \(\mathcal{P}_{\text{unlabelled}}\) can be different and are not assumed to be optimal. Second, many state-only imitation learning algorithms (e.g., Gupta et al. (2017); Torabi et al. (2018a;b); Liu et al. (2018); Sermanet et al. (2018)) permit (reward-free) interactions with the environments similar to their original counterparts (e.g., Ho & Ermon (2016); Kim et al. (2020)). This is not allowed in our offline setup, where the agents are only provided with \(\mathcal{T}_{\text{labelled}}\) and \(\mathcal{T}_{\text{unlabelled}}\).

3.2 Training Pipeline

RL policies trained on low to moderate quality offline trajectories are often sub-optimal, as many of the trajectories might not have high returns and only cover a limited part of the state space. Our goal is to find a way to combine the action labelled trajectories and the unlabelled action-free trajectories, so that the offline agent can exploit structures in the unlabelled data to improve performance.

One natural strategy is to fill in proxy actions for those unlabelled trajectories, and use the proxy-labelled data together with the labelled data as a whole to train an offline RL agent. Since we assume both the labelled and unlabelled trajectories contain the states, we can train an inverse dynamics model (IDM) \(\phi\) that predicts actions using the states. Once we obtain the IDM, we use it to generate the proxy actions for the unlabelled trajectories. Finally, we combine those proxy-labelled trajectories with the labelled trajectories, and train an agent using the offline RL algorithm of choice. Our meta-algorithmic pipeline is summarized in Algorithm 1.

Particularly, we propose a novel stochastic multi-transition
Semi-Supervised Offline Reinforcement Learning with Action-Free Trajectories

Algorithm 1: Semi-supervised offline RL (SS-ORL)

1. **Input:** trajectories $T_{\text{labelled}}$ and $T_{\text{unlabelled}}$, IDM transition size $k$, offline RL algorithm ORL

2. // train a stochastic multi-transition IDM using the labelled data

3. // fill in the proxy actions for the unlabelled data

4. for each trajectory $\tau \in T_{\text{unlabelled}}$ do

5. $\hat{\theta} \leftarrow \arg\min_{\theta} \sum (a_t, s_{t-k}) \in T_{\text{labelled}} [-\log \phi_t(a_t|s_{t-k})]$

6. $T_{\text{proxy}} \leftarrow \tau$ with proxy actions $\{\hat{a}_t\}_{t=1}^{\tau}$ filled in

7. $T_{\text{proxy}} \leftarrow T_{\text{proxy}} \cup \{\tau_{\text{proxy}}\}$

8. $\pi \leftarrow$ policy trained by ORL using dataset $T_{\text{labelled}} \cup T_{\text{proxy}}$

9. **Output:** $\pi$

IDM that incorporates past information to enhance the treatment for stochastic MDPs and non-Markovian behavior policies. Section 3.2.1 discusses the details. Of note, SS-ORL is a multi-stage pipeline, where the IDM is trained only on the labelled data in a single round. There are other possible ways to combine the labelled and unlabelled data. In Section 3.2.2, we discuss several alternative design choices and the key reasons why we do not employ them. Additionally, we present the ablation experiments in Section 4.2.

### 3.2.1 Stochastic Multi-transition IDM

In past work (Pathak et al., 2017; Burda et al., 2019; Henaff et al., 2022), the IDM typically learns to map two subsequent states of the $t$-th transition, $(s_t, s_{t+1})$, to $a_t$. In theory, this is sufficient when the offline dataset is generated by a single Markovian policy in a deterministic environment, see Appendix D for the analysis. However, in practice, the offline dataset might contain trajectories logged from multiple sources.

To provide better treatment for multiple behavior policies, we introduce a multi-transition IDM that predicts the distribution of $a_t$ using the most recent $k + 1$ transitions. More precisely, let $s_{t-k}$ denote the sequence $s_{\min(0,t-k)}, \ldots, s_t, s_{t+1}$. We model $P(a_t|s_{t-k})$ as a multi-variate Gaussian with a diagonal covariance matrix:

$$a_t \sim \mathcal{N}(\mu_0(s_{t-k}), \Sigma_0(s_{t-k})).$$  \hspace{1cm} (1)

Let $\phi_0(a_t|s_{t-k})$ be the probability density function of $\mathcal{N}(\mu_0(s_{t-k}), \Sigma_0(s_{t-k})).$ Given the labelled trajectories $T_{\text{labelled}}$, we minimize the negative log-likelihood loss $\sum (a_t, s_{t-k}) \in T_{\text{labelled}} [-\log \phi_0(a_t|s_{t-k})]$. We call $k$ the transition size parameter. Note that the standard IDM which predicts $a_t$ from $(s_t, s_{t+1})$ under the $\ell_2$ loss, is a special case subsumed by our model: it is equivalent to the case $k = 0$ and the diagonal entries of $\Sigma_0$ (i.e., the variances of each action dimension) are all the same.

In essence, we approximate $p(a_t|s_{t+1}, \ldots, s_1)$ by $p(a_t|s_{t-k})$, and choosing $k > 0$ allows us to take past state information into account. Meanwhile, the theory also indicates that incorporating future states like $s_{t+k}$ would not help to predict $a_t$ (see the analysis in Appendix D for details). For all the experiments in this paper, we use $k = 1$. We ablate this design choice in Section 4.2. Moreover, our IDM naturally extends to non-Markovian policies and stochastic MDPs. This is beyond the scope of this paper, but we consider them as potential directions for future work.

#### 3.2.2 Alternative Design Choices

**Training without Proxy Labelling** SS-ORL fills in proxy actions for the unlabelled trajectories before training the agent. There, the policy learning task is defined on the combined dataset of the labelled and unlabelled data. An alternative approach is to only use the labelled data to define the policy learning task, but create certain auxiliary tasks using the unlabelled data. These auxiliary tasks do not depend on actions, so that proxy-labelling is not needed. Multitask learning approaches can be employed to train an agent that solves those tasks together. For example, Reed et al. (2022) train a generalist agent that processes diverse sequences with a single transformer model. In a similar vein, we consider DT-Joint, a variant of DT, that trains on both labelled and unlabelled data simultaneously. In a nutshell, DT-Joint predicts actions for the labelled trajectories, and states and rewards for both labelled and unlabelled trajectories. See Appendix F for the implementation details. Nonetheless, our ablation experiment in Section 4.2 shows that SS-ORL significantly outperforms DT-Joint.

**Self-Training for the IDM** The annotation process in SS-ORL, which involves training an IDM on the labelled data and generating proxy actions for the unlabelled trajectories, is similar to one step of self-training (Fralick, 1967, Cf. Section 2), one commonly used approach in standard semi-supervised learning. However, a key difference is that we do not retrain the IDM but directly move to the next stage of training the agent using the combined data. There are a few reasons that we do not employ self-training for the IDM. First, it is computationally expensive to execute multiple rounds of training. More importantly, our end goal is to obtain a downstream policy with improved performance via utilizing the proxy-labelled data. As a baseline, we consider self-training for the IDM, where after each training round we add the proxy-labelled data with low predictive uncertainties into the training set for the next round. Empirically, we found that this variant underperforms our approach. See Section 4.2 and Appendix E for more details.
4 Experiments

Our main objectives are to answer four sets of questions:

Q1. How close can SS-ORL agents match the performance of fully supervised offline RL agents, especially when only a small subset of trajectories is labelled?

Q2. How do the SS-ORL agents perform under different design choices for training the IDM, or even avoiding proxy-labelling completely?

Q3. How does the performance of SS-ORL agents vary as a function of the size and quality of the labelled and unlabelled datasets?

Q4. Do different offline RL methods respond differently to various setups of the dataset size and quality?

We focus on two Gym locomotion tasks, hopper and walker, with the v2 medium-expert, medium and medium-replay datasets from the D4RL benchmark (Fu et al., 2020). Due to space constraints, the results on medium and medium-replay datasets are deferred to Appendix C. We respond to the above questions in Section 4.1, 4.2, 4.3 and 4.4, respectively. We also include additional experiments on the maze2d environments in Appendix H. For all experiments, we train 5 instances of each method with different seeds, and for each instance we roll out 30 evaluation trajectories.

4.1 Main Evaluation (Q1)

Data Setup We subsample 10% of the total offline trajectories whose returns are from the bottom q% as the labelled trajectories, 10 ≤ q ≤ 100. The actions of the remaining trajectories are discarded to create the unlabelled ones. We refer to this setup as the coupled setup, since the labelled data distribution \( P_{\text{labelled}} \) and the unlabelled data distribution \( P_{\text{unlabelled}} \) will change simultaneously as we vary the value of \( q \). As \( q \) increases, the labelled data quality increases and the distributions \( P_{\text{labelled}} \) and \( P_{\text{unlabelled}} \) become closer. When \( q = 100 \), our setup is equivalent to sampling the labelled trajectories uniformly and \( P_{\text{labelled}} = P_{\text{unlabelled}} \). Note that under our setup, we always have 10% trajectories labelled and 90% unlabelled, and the total amount of data used to train the offline RL agent is the same as the original offline dataset. This allows for easy comparison with results under the standard, fully labelled setup. In Section 4.3, we will decouple \( P_{\text{labelled}} \) and \( P_{\text{unlabelled}} \) for an in-depth understanding of their individual influences on the SS-ORL agents.

Inverse Dynamics Model We train an IDM as described in Section 3 with \( k = 1 \). That is, the IDM predicts \( a_t \) using 3 consecutive states: \( s_{t-1}, s_t \) and \( s_{t+1} \), where the mean and the covariance matrix are predicted by two independent multilayer perceptrons (MLPs), each containing two hidden layers and 1024 hidden units per layer. To prevent overfitting, we randomly sample 10% of the labelled trajectories as the validation set, and use the IDM that yields the best validation error within 100k iterations.

Offline RL Methods We instantiate Algorithm 1 with DT, CQL and TD3BC as the underlying offline RL methods. DT is a recently proposed conditional behaviour cloning (BC) method that uses sequence modelling tools to model the trajectories. CQL is a representative value-based offline RL method. TD3BC is a hybrid method which adds a BC term to regularize the Q-learning updates. We refer to these instantiations as SS-DT, SS-CQL and SS-TD3BC, respectively. See Appendix A for the implementation details.

Results We compare the performance of the SS-ORL agents with corresponding baseline and oracle agents. The baseline agents are trained on the labelled trajectories only, and the oracle agents are trained on the full offline dataset with complete action labels. Intuitively, the performance of the baseline and the oracle agents can be considered as the (estimated) lower and upper bounds for the performance of the SS-ORL agents. We consider 6 different values of \( q \): 10, 30, 50, 70, 90 and 100, and we report the average return and standard deviation after 200k iterations. Figure 4.1 plots the results on the medium-expert datasets. On both datasets, the SS-ORL agents consistently improve upon the baselines. Remarkably, even when the labelled data quality is low, the SS-ORL agents are able to obtain decent returns. As \( q \) increases, the performance of the SS-ORL agents also keeps increasing and finally matches the performance of the oracle agents.

To quantitatively measure how a SS-ORL agent tracks the performance of the corresponding oracle agent, we define the relative performance gap of SS-ORL agents as

\[
\frac{\text{Perf}(\text{Oracle-ORL}) - \text{Perf}(\text{SS-ORL})}{\text{Perf}(\text{Oracle-ORL})},
\]

and similarly for the baseline agents. Figure 4.2 plots the average relative performance gap of these agents. Compared with the baselines, the SS-ORL agents notably reduce the relative performance gap.

Our results generalize to even fewer percentage of labelled data. Figure 4.3 plots the relative performance gap of the agents trained on walker-medium-expert datasets, when only 1% of the total trajectories are labelled. See Appendix C.3 for more experiments. Similar observations can be found in the results of medium and medium-replay datasets, see Figure C.1 and C.2.

4.2 Comparison with Alternative Design Choices (Q2)

Training without Proxy-Labelling Figure 4.4 plots the performance of DT-Joint and the SS-ORL agents on the hopper-medium-expert dataset, using the coupled setup as in Section 4.1. Since DT-Joint is a variant of DT, the left panel compares DT-Joint with SS-DT as
Semi-Supervised Offline Reinforcement Learning with Action-Free Trajectories

Figure 4.1: Return (average and standard deviation) of SS-ORL agents trained on the D4RL medium-expert datasets. The SS-ORL agents are able to utilize the unlabelled data to improve their performance upon the baselines and even match the performance of the oracle agents.

Figure 4.2: Relative performance gap of SS-ORL agents and corresponding baselines on hopper and walker-medium-expert datasets.

Figure 4.3: Relative performance gap of SS-ORL agents and corresponding baselines with 1% labelled trajectories.

well as the DT baseline and the DT oracle. DT-Joint only marginally outperforms the DT baseline and performs significantly worse than SS-DT. In addition, the right panel shows that SS-CQL, SS-DT and SS-TD3BC all perform much better than DT-Joint. The implementation details of DT-Joint can be found in Appendix F.

Self-Training for the IDM We consider a variant of SS-ORL where self-training is used to train the IDM. Recall that self-training involves an initial training round using only the labelled data, followed by multiple additional rounds using the augmented training sets. After each training round, we need to measure the uncertainties of our action predictions and add the most ones into the training set. To do this, we use the ensemble based method (Lakshminarayanan et al., 2017) where we train $m$ independent stochastic IDMs. We model the action distribution as the mixture of those $m$ estimated distributions. The whole self-training algorithm is presented in Algorithm 2 in Appendix E.

We compare SS-CQL, SS-DT with their self-training variant on the walker-medium-expert datasets, using IDM with $k = 1$. All the hyperparameters and the architecture are the same. We have tested the variant with ensemble size 2 and 3, and with 3 and 5 augmentation rounds. As before, we use the coupled setup with 6 different $q$ varying between 10 and 100. To take account of different models and different data setups, we report the 95% stratified bootstrap confidence intervals (CIs) of the interquartile mean (IQM)\(^1\) of the return for all these cases and training instances (Agarwal et al., 2021). We use 50000 bootstrap replications to generate the CIs. Compared with the other statistics like the mean or the median, the IQM is both robust to outliers and also a good representative of the overall performance. The stratified bootstrapping is a handy tool to obtain CIs with decent coverage rate, even if one only has a small number of training instances per setup. We refer the readers to Agarwal et al. (2021) for the complete introduction. Figure 4.5 plots the 95% bootstrap CIs of the IQM return across all the setups. Our approach notably outperforms the other variants.

It is intriguing to investigate the MSE of action predictions for different IDMs. Figure 4.6 shows that our IDM

\(^1\)The interquartile mean of a list of sorted numbers is the mean of the middle 50% numbers.
Figure 4.5: The 95% bootstrap CIs of the IQM return obtained by the SS-ORL agents and the variants with self-training IDMs.

Figure 4.6: The action prediction MSE of different IDMs.

Figure 4.7: The 95% bootstrap CIs of the IQM return, when the labelled data is of low or moderate quality.

Figure 4.8: The 95% bootstrap CIs of the IQM return of the SS-ORL agents with varying labelled data quality.

Figure 4.9: The 95% bootstrap CIs of the IQM return of the SS-ORL agents with varying unlabelled data quality.

Figure 4.10: The 95% bootstrap CIs of the IQM return of the SS-ORL agents with varying labelled and unlabelled data quality.

We conduct experiments to investigate the performance of SS-ORL in variety of data settings. To enable a systematic study, we depart from the coupled setup in Section 4.1 and consider a decoupling of $P_{\text{labelled}}$ and $P_{\text{unlabelled}}$. We will vary four configurable values: the quality and size of both the labelled and unlabelled trajectories, individually while keeping the other values fixed. We examine how the performance of the SS-ORL agents change with these variations.

### 4.3 Alibation Study for Data-Centric Properties (Q3)

Quality of Labelled Data We divide the offline trajectories into 3 groups, whose returns are the bottom 0% to 33%, 33% to 67%, and 67% to 100%, respectively. We refer to them as Low, Medium, and High groups. We evaluate the performance of SS-ORL when the labelled trajectories are sampled from three different groups: Low, Med, and High. To account for different environment, offline RL methods, and the unlabelled data qualities, we consider a total of 12 cases that cover:

- 2 datasets hopper-medium-expert and walker-medium-expert,
- 2 agents SS-CQL and SS-DT, and
- 3 quality setups where the unlabelled trajectories are sam-
Both the number of labelled and unlabelled trajectories are set to be 10% of the total number of offline trajectories. Figure 4.9 report the 95% bootstrap CIs of the IQM return for all the 12 cases and 5 training instances per case. Clearly, as the labelled data quality goes up, the performance of SS-ORL significantly increases by large margins.

**Quality of Unlabelled Data** Similar to the above experiment, we sample the unlabelled trajectories from one of the three groups, and train the SS-ORL agents under 12 different cases where the labelled data quality varies. Figure 4.10 reports the 95% bootstrap CIs of the IQM return. The performance of SS-ORL agents increases as the unlabelled data quality increases, and using high quality unlabelled data significantly outperforms the other two cases.

**Size of Labelled Data** We vary the number of labelled trajectories as 10%, 25%, and 50% of the offline dataset size, while the number of unlabelled trajectories is fixed to be 10%. We train SS-CQL and SS-DT on the walker-medium-expert dataset under 9 data quality setups, where the labelled and unlabelled trajectories are respectively sampled from Low, Med, and High groups. Figure 4.11 plots the CIs of the IQM return. Specifically, we consider the results aggregated over all the cases, and also for each individual labelled data quality setup. For all these cases, the performance of both SS-CQL and SS-DT remain relatively consistent regardless of the number of labelled trajectories. The evaluation performance of SS-CQL and SS-DT over the course of training for each individual environment and data setup, can be found in Figure G.1.

**Size of Unlabelled Data** As before, we vary the percentage of unlabelled trajectories as 10%, 25%, and 50%, while fixing the labelled data percentage to be 10%. We use the same data quality setups as in the previous experiment. Figure 4.12 reports the 95% bootstrap CIs of the IQM return. Interestingly, we found that SS-DT and SS-CQL respond slightly differently. SS-CQL is relatively insensitive to changes in the size of the unlabelled data, as is SS-DT when the labelled data quality is low or moderate. However, when labelled data is of high quality, the performance of SS-DT deteriorates as the unlabelled data size increases. To gain a better understanding of this phenomenon, we investigate the performance for SS-DT for each of the 9 data quality setups. As shown in Figure G.2a, when the labelled data is of high quality but the unlabelled data is of lower quality, growing the unlabelled data size harms the performance. Our intuition is that, in these cases, the combined dataset will have lower quality than the labelled dataset, and supervised learning approaches like DT can be sensitive to this. More detailed can be found in Figure G.2.

4.4 The Choice of Offline RL Algorithm (Q4)

For a chosen offline RL method, the relative performance gap of the SS-ORL agents instantiated with different offline RL methods.
5 Conclusion

We proposed a novel semi-supervised setup for offline RL where we have access to trajectories with and without action information. For this setting, we introduced a simple multi-stage meta-algorithmic pipeline. Our experiments identified key properties that enable the agents to leverage unlabelled data and show that near-optimal learning can be done with only 10% of the actions labelled for low-to-moderate quality trajectories. Our work is a step towards creating intelligent agents that can learn from diverse types of auxiliary demonstrations like online videos, and it would be interesting to study other heterogeneous data setups for offline RL in the future, including reward-free or pure state-only settings.

Acknowledgement

The authors thank Maryam Fazel-Zarandi, Chi Jin, Mike Rabbat, Aravind Rajeswaran, Yuandong Tian, Lin Xiao, Denis Yarats, Amy Zhang and Dinghuai Zhang for insightful discussions.

References


Ijspeert, A. J., Nakanishi, J., and Schaal, S. Movement imitation with nonlinear dynamical systems in humanoid


A Experiment Details

In this section, we provide more details about our experiments. For all the offline RL methods we consider, we use our own implementations adopted from the following codebases:

DT  https://github.com/facebookresearch/online-dt
TD3BC  https://github.com/sfujim/TD3_BC
CQL  https://github.com/scottemmons/youungs-cql

We use the stochastic DT proposed by Zheng et al. (2022). For offline RL, its performance is similar to the deterministic DT (Chen et al., 2021). The policy parameter is optimized by the LAMB optimizer (You et al., 2019) with $\varepsilon = 10^{-8}$. The log-temperature parameter is optimized by the Adam optimizer (Kingma & Ba, 2014). The architecture and other hyperparameters are listed in Table A.1. For TD3BC, we optimize both the critic and actor parameters by the Adam optimizer. The complete hyperparameters are listed in Table A.2. For CQL, we also use the Adam optimizer to optimize the critic, actor and the log-temperature parameters. The architecture of critic and actor networks and the other hyperparameters are listed in Table A.3. We use batch size 256 and context length 20 for DT, where each batch contains 5120 states. Correspondingly, we use batch size 5120 for CQL and TD3BC.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of layers</td>
<td>4</td>
</tr>
<tr>
<td>number of attention heads</td>
<td>4</td>
</tr>
<tr>
<td>embedding dimension</td>
<td>512</td>
</tr>
<tr>
<td>context length</td>
<td>20</td>
</tr>
<tr>
<td>dropout</td>
<td>0.1</td>
</tr>
<tr>
<td>activation function</td>
<td>relu</td>
</tr>
<tr>
<td>batch size</td>
<td>256</td>
</tr>
<tr>
<td>learning rate for policy</td>
<td>0.0001</td>
</tr>
<tr>
<td>weight decay for policy</td>
<td>0.001</td>
</tr>
<tr>
<td>learning rate for log-temperature</td>
<td>0.0001</td>
</tr>
<tr>
<td>gradient norm clip</td>
<td>0.25</td>
</tr>
<tr>
<td>learning rate warmup</td>
<td>linear warmup for $10^4$ steps</td>
</tr>
<tr>
<td>target entropy</td>
<td>$-\text{dim}(A)$</td>
</tr>
<tr>
<td>evaluation return-to-go</td>
<td>3600 Hopper</td>
</tr>
<tr>
<td></td>
<td>5000 Walker</td>
</tr>
<tr>
<td></td>
<td>6000 HalfCheetah</td>
</tr>
</tbody>
</table>

Table A.1: The hyperparameters used for DT.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>discount factor</td>
<td>0.99</td>
</tr>
<tr>
<td>target update rate</td>
<td>0.005</td>
</tr>
<tr>
<td>policy noise</td>
<td>0.2</td>
</tr>
<tr>
<td>policy noise clipping</td>
<td>$(-0.5, 0.5)$</td>
</tr>
<tr>
<td>policy update frequency</td>
<td>2</td>
</tr>
<tr>
<td>critic learning rate</td>
<td>0.0003</td>
</tr>
<tr>
<td>critic hidden dim</td>
<td>256</td>
</tr>
<tr>
<td>critic hidden layers</td>
<td>2</td>
</tr>
<tr>
<td>actor learning rate</td>
<td>0.0003</td>
</tr>
<tr>
<td>actor hidden dim</td>
<td>256</td>
</tr>
<tr>
<td>actor hidden layers</td>
<td>2</td>
</tr>
<tr>
<td>activation function</td>
<td>ReLU</td>
</tr>
<tr>
<td>regularization parameter $\alpha$</td>
<td>2.5</td>
</tr>
</tbody>
</table>

Table A.2: The hyperparameters used for TD3BC.
Table A.3: The hyperparameters used for CQL.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>discount factor</td>
<td>0.99</td>
</tr>
<tr>
<td>target update rate</td>
<td>0.005</td>
</tr>
<tr>
<td>critic learning rate</td>
<td>0.0003</td>
</tr>
<tr>
<td>critic hidden dim</td>
<td>256</td>
</tr>
<tr>
<td>critic hidden layers</td>
<td>3</td>
</tr>
<tr>
<td>actor learning rate</td>
<td>0.0001</td>
</tr>
<tr>
<td>actor hidden dim</td>
<td>256</td>
</tr>
<tr>
<td>actor hidden layers</td>
<td>3</td>
</tr>
<tr>
<td>log-temperature learning rate</td>
<td>0.0003</td>
</tr>
<tr>
<td>activation function</td>
<td>ReLU</td>
</tr>
<tr>
<td>number of sampled actions</td>
<td>10</td>
</tr>
<tr>
<td>target entropy</td>
<td>$-\dim(A)$</td>
</tr>
<tr>
<td>minimum Q weight value</td>
<td>5</td>
</tr>
<tr>
<td>Lagrange</td>
<td>False</td>
</tr>
<tr>
<td>Importance Sampling</td>
<td>True</td>
</tr>
</tbody>
</table>

B The Return Distributions of the D4RL Datasets

![Distributions of the normalized returns of the D4RL datasets.](image)

Figure B.1: The distributions of the normalized returns of the D4RL datasets.

C Additional Experiments Under the Coupled Setup

C.1 Experiments on medium and medium-replay and all halfcheetah Datasets

We conduct experiments on the medium and medium-replay datasets of D4RL benchmark for the hopper and walker environments, using the same setup as in Section 4.1. Figure C.1 and C.2 reports the results. For completeness, we also report the results on medium-expert, medium, and medium-replay datasets for the halfcheetah...
environment in Figure C.3. We found relatively suboptimal results for DT on the halfcheetah environment, consistent with prior results in Zheng et al. (2022). The general trend is the same as that in Figure 4.1. We note that the results on the halfcheetah-medium dataset are less informative than the others. This is because the data distributions of halfcheetah-medium is very concentrated, similar to a Gaussian distribution with small variance, see Figure B.1. In such a case, varying the value of \( q \) does not drastically change the labelled data distribution. To verify our hypothesis, we conduct experiments on a subsampled dataset in the next subsection.

Figure C.1: The return (average and standard deviation) of SS-ORL agents trained on the D4RL medium datasets for hopper and walker.

Figure C.2: The return (average and standard deviation) of SS-ORL agents on the D4RL medium-replay datasets for hopper and walker.
C.2 Performance of SS-ORL on a Subsampled Dataset with Wide Return Distribution

One may notice that for the hopper-medium-replay and walker2d-medium-replay datasets, SS-ORL does not catch up with the oracle as quickly as on the other datasets as \( q \) increases. Our intuition is that the return distributions of these two datasets concentrate on extremely low values, as shown in Figure B.1. In our experiments, the labelled trajectories for those two datasets have average return small than 0.1 even when \( q = 70 \). In contrast, the return distributions of the other datasets concentrate on larger values. In contrast, for the other datasets, increasing the value of \( q \) will greatly change the returns of labelled trajectories, see Table C.1.

<table>
<thead>
<tr>
<th>dataset</th>
<th>q=10</th>
<th>q=30</th>
<th>q=50</th>
<th>q=70</th>
<th>q=90</th>
<th>q=100</th>
</tr>
</thead>
<tbody>
<tr>
<td>hopper-medium-replay</td>
<td>0.007</td>
<td>0.022</td>
<td>0.05</td>
<td>0.074</td>
<td>0.109</td>
<td>0.149</td>
</tr>
<tr>
<td>walker2d-medium-replay</td>
<td>-0.002</td>
<td>0.005</td>
<td>0.023</td>
<td>0.048</td>
<td>0.087</td>
<td>0.156</td>
</tr>
<tr>
<td>halfcheetah-medium-replay</td>
<td>0.001</td>
<td>0.092</td>
<td>0.179</td>
<td>0.202</td>
<td>0.269</td>
<td>0.275</td>
</tr>
<tr>
<td>hopper-medium</td>
<td>0.231</td>
<td>0.310</td>
<td>0.355</td>
<td>0.388</td>
<td>0.418</td>
<td>0.443</td>
</tr>
<tr>
<td>walker2d-medium</td>
<td>0.135</td>
<td>0.287</td>
<td>0.44</td>
<td>0.557</td>
<td>0.599</td>
<td>0.618</td>
</tr>
<tr>
<td>halfcheetah-medium</td>
<td>0.361</td>
<td>0.383</td>
<td>0.397</td>
<td>0.396</td>
<td>0.406</td>
<td>0.405</td>
</tr>
<tr>
<td>hopper-medium-expert</td>
<td>0.252</td>
<td>0.341</td>
<td>0.394</td>
<td>0.451</td>
<td>0.594</td>
<td>0.645</td>
</tr>
<tr>
<td>walker2d-medium-expert</td>
<td>0.201</td>
<td>0.469</td>
<td>0.605</td>
<td>0.732</td>
<td>0.791</td>
<td>0.827</td>
</tr>
<tr>
<td>halfcheetah-medium-expert</td>
<td>0.377</td>
<td>0.397</td>
<td>0.405</td>
<td>0.537</td>
<td>0.604</td>
<td>0.638</td>
</tr>
</tbody>
</table>

Table C.1: The average return of the labelled trajectories in our experiments. Results aggregated over 5 seeds.

To demonstrate the performance of SS-ORL on dataset with a more wide return distribution, we consider a subsampled dataset for the walker environment generated as follows.
1. Combine the walker-medium-replay and walker-medium datasets.

2. Let $R_{\text{min}}$ and $R_{\text{max}}$ denote the minimum and maximum return in the dataset. We divide the trajectories into 40 bins, where the maximum returns within each bin are linear spaced between $R_{\text{min}}$ and $R_{\text{max}}$. Let $n_i$ be the number trajectories in bin $i$.

3. We randomly sample 1000 trajectories. To sample a trajectory, we first first sample a bin $i \in [1, \ldots, 40]$ with weights proportional to $1/n_i$, then sample a trajectory uniformly at random from the sampled bin.

Figure C.4 plots the return distribution of the subsampled dataset. It is wide and has 3 modes. We run the same experiments as before on this subsampled dataset, and Figure C.5 plots the results. The general trend is the same as we have found in the above experiments.

![Figure C.4: The density of a randomly subsampled dataset of the walker environment.](image)

![Figure C.5: The return (average and standard deviation) of SS-ORL agents on the subsampled dataset.](image)

C.3 Results on Low Percentages of Labelled Data

We present the results when the number of the labelled trajectories are 1%, 3%, 5%, and 8% of the total offline dataset size. Figure C.6 plots the absolute returns and Figure C.7 plots the relative performance gaps. We observe the same trend as the experiments in Section 4.1.
Figure C.6: The return (average and standard deviation) of SS-ORL agents trained on the walker-medium-expert dataset, when 1%, 3%, 5% and 8% of the offline trajectories are labelled.
D Analysis of the Multi-Transition Inverse Dynamics Model

Given all the past states, we can write

\[
p(a_t|s_{t+1}, \ldots, s_1) = \frac{p(a_t, s_{t+1}, \ldots, s_1)}{p(s_{t+1}, \ldots, s_1)} = \frac{p(s_{t+1}|a_t, s_t, \ldots, s_1)p(a_t|s_{t+1}, \ldots, s_1)}{p(s_{t+1}|s_t, \ldots, s_1)} = \frac{p(s_{t+1}|a_t, s_t)p(a_t|s_{t+1}, \ldots, s_1)}{p(s_{t+1}|s_t, \ldots, s_1)} = \frac{p(s_{t+1}|a_t, s_t)p(a_t|s_{t+1}, \ldots, s_1)}{\int_{a \in A} p(s_{t+1}|a_t, s_t)p(a_t|s_{t+1}, \ldots, s_1)},
\]

where the last two lines follow from the the Markovian transition property \( p(s_{t+1}|a_t, s_t, \ldots, s_1) = p(s_{t+1}|a_t, s_t) \) inherent to a Markov Decision Process.

Let \( \beta \) denote the behavior policy. If \( \beta \) is Markovian, then we have \( p(a_t|s_t, \ldots, s_1) = \beta(a_t|s_t) \) and it holds that

\[
p(a_t|s_{t+1}, \ldots, s_1) = \frac{p(s_{t+1}|a_t, s_t)\beta(a_t|s_t)}{\int_{a \in A} p(s_{t+1}|a_t, s_t)\beta(a_t|s_t)} = p(a_t|s_{t+1}, s_t).
\]
Similarly, if \( \beta \) is non-Markovian and takes account of the previous \( k \) states as well, we have

\[
p(a_t|s_{t+1}, \ldots, s_1) = p(a_t|s_{t+1}, s_t, \ldots, s_{t-k}).
\]

While the past work commonly models \( p(a_t|s_{t+1}, s_t) \) (Pathak et al., 2017; Burda et al., 2019; Henaff et al., 2022), in practice, the offline dataset might contain trajectories generated by multiple behavior policies, and it is unknown if any of them is Markovian.

Our formulation has considered these situations: 1) the behavior policy is non-Markovian, 2) there are multiple behavior policies, 3) and the environment is stochastic. First, from the above derivation, we can see that choosing \( k > 0 \) allows us to take into account past information before timestep \( t \), which naturally copes with non-Markovian policies. Second, for the case where there are multiple Markovian behavior policies (we assume the behavior policies are Markovian for simplicity), we believe it is easier to infer the actual behavior policy by a sequence of past states rather than a single one. Last, the past work usually predicts \( a_t \) via a deterministic function of \( (s_t, s_{t+1}) \), which implicitly assumes a deterministic environment. In the contrary, our approach has stochasticity, which can potentially better cope with the stochastic environment. Due to the practical limitation of testing environment and dataset, our experiments only show that the multi-transition IDM outperforms the classic one when the datasets are generated by multiple behavior policies, see Section 4.2. We leave whether the multi-transition IDM provides a better solution to non-Markovian policies and stochastic environments an open question and consider it as one of future work.

A natural question to ask is whether we should incorporate any future states such as \( s_{t+2} \). Figure D.1 depicts the graphical model of the state transitions under a MDP. It is easy to see that given \( s_t \) and \( s_{t+1} \), \( a_t \) is independent of \( s_{t+2} \) and all the future states (Koller & Friedman, 2009).

![Figure D.1: Graphical model of a Markovian behavior policy (curved) within the transition dynamics of an MDP (straight). For non-Markovian behavioral policies, we will have additional arrows from \( s_{t-k} \) to \( a_t \) for \( k > 0 \).](image)

In the experiments in Section 4.2, we empirically verify that including future states do not help predicting the actions. Meanwhile, the transition window size \( k \) is a hyperparameter we need to choose. For all our experiments, we use \( k = 1 \) and hence incorporate information about \( s_{t-1} \) as well. We ablate this choice in Section 4.2, see Figure 4.8.
E Self-Training for IDM

We present the self-training algorithm used to train the IDM in Algorithm 2. In each training round, we randomly sample 10% of the training data as the validation set. During the training of each individual IDM, we select the model that yields the best validation error in 100k iterations.

Algorithm 2: Self-Training for the Inverse Dynamics Model

\begin{enumerate}
\item \textbf{Input}: labelled data $D_{\text{labelled}}$, unlabelled data $D_{\text{unlabelled}}$, IDM transition size $k$, ensemble size $m$, number of augmentation rounds $N$
\item // initialize the training set
\item $D \leftarrow D_{\text{labelled}}$
\item // train $m$ independent IDMs using the labelled data under the randomness of initialization and data shuffling
\item $\hat{\theta}_i \leftarrow \arg\min_{\theta} \sum_{(s_t, a_{t-k})} \in D \left[- \log \phi_\theta (a_t | s_{t-k}), i \in [m]\right]$
\item // compute the augmentation size
\item $n_{\text{aug}} \leftarrow |D_{\text{unlabelled}}| / N$
\item for round $1, \ldots, N$ do
\item \hspace{1em} // compute the estimation uncertainty
\item \hspace{2em} for every $(a_t, s_{t-k}) \in D_{\text{unlabelled}}$ do
\item \hspace{3em} $\nu_t \leftarrow$ variance of the Gaussian mixture $\frac{1}{m} \sum_{i=1}^{m} N \left( \mu_{\hat{\theta}_i} (s_{t-k}), \Sigma_{\hat{\theta}_i} (s_{t-k}) \right)$
\item \hspace{1em} // move examples with lowest uncertainties into the training set
\item \hspace{2em} $D_{\text{subset}} \leftarrow \{(a_t, s_{t-k}) | \nu_t \text{ among the lowest } n_{\text{aug}} \text{ in } D_{\text{unlabelled}}\}$
\item \hspace{1em} $D \leftarrow D \cup D_{\text{subset}}$
\item \hspace{1em} $D_{\text{unlabelled}} \leftarrow D_{\text{unlabelled}} \setminus D_{\text{subset}}$
\item \hspace{1em} // train IDMs again
\item \hspace{2em} $\hat{\theta}_i \leftarrow \arg\min_{\theta} \sum_{(a_t, s_{t-k})} \in D \left[- \log \phi_\theta (a_t | s_{t-k}), i \in [m]\right]$
\item \hspace{1em} Output: $\hat{\theta}_1, \ldots, \hat{\theta}_m$
\end{enumerate}

F Implementation Details of DT–Joint

Inspired by GATO, the multi-task and multi-modal generalist agent proposed by Reed et al. (2022), we consider DT–Joint, a variant of DT that can incorporate the unlabelled data into policy training. DT–Joint is trained on the labelled and unlabelled data simultaneously. The implementation details are:

- We form the same input sequence as DT, where we fill in zeros for the missing actions for unlabelled trajectories.
- For the labelled trajectories, DT–Joint predicts the actions, states and rewards; for the unlabelled ones, DT–Joint only predicts the states and rewards.
- We use the stochastic policy as in online decision transformer (Zheng et al., 2022) to predict the actions.
- We use deterministic predictors for the states and rewards, which are single linear layers built on top of the Transformer outputs.

Let $g_t = \sum_{t'=t}^{t+|\mathcal{g}|} r_{t'}$ be the return-to-go of a trajectory $\tau$ at timestep $t$. Let $H^P_{\text{labelled}}$ denotes the policy entropy included on the labelled data distribution. For simplicity, we assume the context length of DT–Joint is 1, and Equation (6) shows the training objective of DT–Joint. (We refer the readers to Zheng et al. (2022) for the formulation with a general context length and more details.)

\[
\begin{align*}
\min_{\theta} \quad & \mathbb{E}_{(s_t, a_t, r_t, g_t) \sim P_{\text{labelled}}} \left\{ - \log \pi(a_t | s_t, g_t, \theta) + \lambda_s \| s_t - \hat{s}_t(\theta) \|^2_2 + \lambda_r \| r_t - \hat{r}_t(\theta) \|^2_2 \right\} \\
& + \mathbb{E}_{(s_t, a_t, r_t, g_t) \sim P_{\text{unlabelled}}} \left\{ \lambda_s \| s_t - \hat{s}_t(\theta) \|^2_2 + \lambda_r \| r_t - \hat{r}_t(\theta) \|^2_2 \right\} \\
\text{s.t.} \quad & H^P_{\text{labelled}} [a | s, g] \geq \nu
\end{align*}
\]
Figure F.1: The 95% stratified bootstrap CIs of the interquartile mean of the returns obtained by DT-Joint agents, with different combinations of regularization parameters.

The constant $\nu$, $\lambda_s$, and $\lambda_r$ are prefixed hyper-parameters, where $\nu$ is the target policy entropy, and $\lambda_s$ and $\lambda_r$ are regularization parameters used to balance the losses for actions, states, and rewards. We use $\nu = -\dim(A)$ as for DT (see Appendix A). To choose the regularization parameters $\lambda_s$ and $\lambda_r$ for DT-Joint, we test 16 combinations where $\lambda_s$ and $\lambda_r$ are 1.0, 0.1, 0.01 and 0.001 respectively. We run experiments as in Section 4.1 for $q = 10, 30, 50, 70, 90, 100$, and compute the confidence intervals for the aggregated results. Figure F.1 shows that $\lambda_s = 0.01$ and $\lambda_r = 0.1$ yield the best performance, and we use them in our experiments for Figure 4.4.
G Influences of the Labelled and Unlabelled Data Size

Figure G.1 plots the average return of SS-DT and SS-CQL when we vary the number of labelled trajectories while fixing the number of unlabelled trajectories. As described in Section 4.3, we consider 9 data setups where the labelled and unlabelled trajectories are sampled from Low, Medium and High groups. In all the plots, L x H denotes the setup where the labelled data are sampled from Low group and the unlabelled data are sampled from High group. Similarly, Figure G.2 plots the results when we vary the number of unlabelled trajectories, while the number of labelled ones is fixed.

Figure G.1: The return (average and standard deviation) of SS-DT and SS-CQL agents trained on the walker-medium-expert datasets with different sizes of labelled data. The unlabelled data size is fixed to be 10% of the offline dataset size. Results aggregated over 5 instances with different seeds.
Figure G.2: The return (average and standard deviation) of SS-DT and SS-CQL agents trained on the walker-medium-expert datasets with different sizes of unlabelled data. The labelled data size is fixed to be 10% of the offline dataset size. Results aggregated over 5 instances with different seeds.
H  Additional Experiments on the Maze2d Environment

The maze2d environment involves moving force-actuated ball to a fixed target location. The observation consists of the location and velocities, and the reward is the negative exponentiated distance to the target location.

We conduct experiments on four offline dataset for the maze2d environments, each corresponds to a different map: maze2d-open-dense-v0, maze2d-umaze-dense-v1, maze2d-medium-dense-v1, and maze2d-large-dense-v1. Figure H.1 plots the normalized return distributions of these four datasets. The return distribution of maze2d-open-dense-v0 is widely spread, while the others are heavily skewed towards the low return values. Note that many of the trajectories’ normalized returns are below zero.

![Figure H.1: The distributions of the normalized returns of the maze2d datasets.](image)

We train SS-ORL agents instantiated with TD3BC, under the coupled setup as in Section 4.1. We use learning rate value 0.0001 for both actors and critics, which is smaller than what we used for locomotion tasks. All the other hyperparameters are the same as described in Appendix A.

Figure H.2 plots the results. The general trend is similar to what we have seen in previous experiments for locomotion tasks.

![Figure H.2: The return (average and standard deviation) of SS-TD3BC agents trained on the maze2d dataset.](image)