IQL-TD-MPC: Implicit Q-Learning for Hierarchical Model Predictive Control

Rohan Chitnis [*1] **Yingchen Xu** [*1 2] **Bobak Hashemi** [1] **Lucas Lehnert** [1] **Urun Dogan** [1] **Zheqing Zhu** [1] **Olivier Delalleau** [1]

Abstract

Model-based reinforcement learning (RL) has shown great promise due to its sample efficiency, but still struggles with long-horizon sparse-reward tasks, especially in offline settings where the agent learns from a fixed dataset. We hypothesize that model-based RL agents struggle in these environments due to a lack of long-term planning capabilities, and that planning in a temporally abstract model of the environment can alleviate this issue. In this paper, we make two key contributions: 1) we introduce an offline model-based RL algorithm, IQL-TD-MPC, that extends the state-of-the-art Temporal Difference Learning for Model Predictive Control (TD-MPC) with Implicit Q-Learning (IQL); 2) we propose to use IQL-TD-MPC as a Manager in a hierarchical setting with *any* off-the-shelf offline RL algorithm as a Worker. More specifically, we pre-train a temporally abstract IQL-TD-MPC Manager to predict "intent embeddings", which roughly correspond to subgoals, via planning. We empirically show that augmenting state representations with intent embeddings generated by an IQL-TD-MPC manager significantly improves off-the-shelf offline RL agents' performance on some of the most challenging D4RL benchmark tasks. For instance, the offline RL algorithms AWAC, TD3-BC, DT, and CQL all get zero or near-zero normalized evaluation scores on the medium and large antmaze tasks, while our modification gives an average score over 40.

[*]Equal contribution [1]Meta AI [2]University College London. Correspondence to: Rohan Chitnis <ronuchit@meta.com>.

Workshop on New Frontiers in Learning, Control, and Dynamical Systems at the International Conference on Machine Learning (ICML), Honolulu, Hawaii, USA, 2023.

1. Introduction

Model-based reinforcement learning (RL), in which the agent learns a predictive model of the environment and uses it to plan and/or train policies (Ha and Schmidhuber, 2018; Hafner et al., 2019; Schrittwieser et al., 2020), has shown great promise due to its sample efficiency compared to its model-free counterpart (Ye et al., 2021; Micheli et al., 2022). Most prior work focuses on learning single-step models of the world, with which planning can be computationally expensive and model prediction errors may compound over long horizons (Argenson and Dulac-Arnold, 2021; Clavera et al., 2020). As a result, model-based RL still struggles with long-horizon sparse-reward tasks, whereas some evidence suggests that humans are able to combine spatial and temporal abstractions to plan efficiently over long horizons (Botvinick and Weinstein, 2014). Modeling the world at a higher level of abstraction can enable predicting long-term future outcomes more accurately and efficiently.

The challenge of long-horizon sparse-reward tasks is particularly prominent in offline RL, where an agent must learn from a fixed dataset rather than from exploring an environment (Levine et al., 2020; Prudencio et al., 2023; Lange et al., 2012; Ernst et al., 2005). The offline setting is key to training RL agents safely, but poses unique challenges such as value mis-estimation (Levine et al., 2020).

In this paper, we study offline model-based RL, and hypothesize that planning in a learned temporally abstract model of the environment can produce significant improvements over "flat" algorithms that do not use temporal abstraction. Our paper makes two key contributions:

- **Section 3**: We propose IQL-TD-MPC, an offline model-based RL algorithm that combines the state-of-the-art online RL algorithm Temporal Difference Learning for Model Predictive Control (TD-MPC) (Hansen et al., 2022) with the popular offline RL algorithm Implicit Q-Learning (IQL) (Kostrikov et al., 2022). This combination requires several non-trivial design decisions.
- **Section 4**: We show how to use IQL-TD-MPC as a Manager in a temporally abstracted hierarchical setting with *any* off-the-shelf offline RL algorithm as a Worker. To achieve this hierarchy, we pre-train an IQL-TD-MPC

Manager to output "intent embeddings" via MPC planning, then during Worker training and evaluation, simply concatenate these embeddings to the environment states. These intent embeddings roughly correspond to subgoals[1] set k steps ahead, thanks to the coarser timescale used when training the Manager. A benefit of this concatenation strategy is its simplicity: it does not require modifying Worker training algorithms or losses.

See Fig. 1 for an overview of our framework. Experimentally, we study the popular D4RL benchmark (Fu et al., 2020). We begin by showing that IQL-TD-MPC is far superior to vanilla TD-MPC and is on par with several other popular offline RL algorithms. Then, we show the significant benefits of our proposed hierarchical framework. For instance, the well-established offline RL algorithms Advantage Weighted Actor Critic (AWAC) (Nair et al., 2020), Twin Delayed DDPG Behavioral Cloning (TD3-BC) (Fujimoto and Gu, 2021), Decision Transformer (DT) (Chen et al., 2021), and Conservative Q-Learning (CQL) (Kumar et al., 2020) all get zero or near-zero normalized evaluation score on the medium and large antmaze variants of D4RL, whereas they obtain an average score of over 40 when used as Workers in our hierarchical framework. Despite the superior performance of our approach on the maze navigation tasks, our empirical analysis shows that such hierarchical reasoning can be harmful in fine-grained locomotion tasks like the D4RL half-cheetah. Overall, our results suggest that model-based planning in a temporal abstraction of the environment can be a general-purpose solution to boost the performance of many different offline RL algorithms, on complex tasks that benefit from higher-level reasoning. Video results are available at `https://sites.google.com/view/iql-td-mpc`.

2. Preliminaries

In this section, we briefly recap the offline RL setting, then provide a detailed review of TD-MPC.

2.1. Markov Decision Processes and Offline Reinforcement Learning

We consider the standard infinite-horizon Markov Decision Process (MDP) (Puterman, 1990) setting with continuous states and actions, defined by a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma, p_0)$ where $\mathcal{S} \subseteq \mathbb{R}^n$ is the state space, $\mathcal{A} \subseteq \mathbb{R}^m$ is the action space, $P(s' \mid s, a)$ is the transition probability distribution function, $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function, $\gamma \in (0, 1)$ is the discount factor, and $p_0(s)$ is the initial state distribution function. The reinforcement learning (RL) objective is to find a policy $\pi(a \mid s)$ that maximizes the expected infinite sum of discounted rewards: $\mathbb{E}_{s_0 \sim p_0, a_t \sim \pi(\cdot|s_t), s_{t+1} \sim P(\cdot|s_t, a_t)} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$.

In offline RL (Levine et al., 2020; Prudencio et al., 2023), the agent learns from a fixed dataset rather than collecting its own data in the environment. One key challenge is dealing with out-of-distribution actions: if the learned policy samples actions for a given state that were not seen in the training set, the model may mis-estimate the value of these actions, leading to poor behavior. Imitation learning methods like Behavioral Cloning (BC) sidestep this issue by mimicking the behavior policy used to generate the dataset, but may perform sub-optimally with non-expert data (Hussein et al., 2017).

2.2. Temporal Difference Model Predictive Control (TD-MPC)

Our work builds on TD-MPC (Hansen et al., 2022), an algorithm that combines planning in a latent space using Model Predictive Control (MPC) with actor-critic Temporal Difference (TD) learning. The components of TD-MPC (with θ denoting the set of all parameters) are the following:

- An encoder $h_\theta : \mathcal{S} \to \mathbb{R}^d$ mapping a state s to its latent representation $z = h_\theta(s)$.
- A forward dynamics model $f_\theta : \mathbb{R}^d \times \mathcal{A} \to \mathbb{R}^d$, predicting the next latent state $\hat{z}' = f_\theta(z, a)$.
- A reward predictor $R_\theta : \mathbb{R}^d \times \mathcal{A} \to \mathbb{R}$ computing expected rewards $\hat{r} = R_\theta(z, a)$.
- A policy $\pi_\theta : \mathbb{R}^d \times \mathcal{A} \to \mathbb{R}^+$ used to sample actions $a \sim \pi_\theta(\cdot \mid z)$.
- A critic $Q_\theta : \mathbb{R}^d \times \mathcal{A} \to \mathbb{R}$ computing state-action values $Q_\theta(z, a)$ that estimate Q-values under π_θ: $Q_\theta(h_\theta(s), a) \simeq Q^{\pi_\theta}(s, a) \triangleq \mathbb{E}_{\pi_\theta}[\sum_{t \geq 0} \gamma^t R_\theta(s_t, a_t) \mid s_0 = s, a_0 = a)]$.

The parameters θ of these components are learned by minimizing several losses over sub-trajectories $(s_0, a_0, r_1, s_1, a_1, \ldots, r_T, s_T)$ sampled from the replay buffer, where T is the horizon:

- A critic loss based on the TD error, $\mathcal{L}_Q = (Q_\theta(\hat{z}_t, a_t) - [r_{t+1} + \gamma Q_{\theta^-}(z_{t+1}, \pi_\theta(z_{t+1}))])^2$, where we denote by $\pi_\theta(z)$ a sample from $\pi_\theta(\cdot \mid z)$ and $\hat{z}_t = f_\theta(\hat{z}_{t-1}, a_{t-1})$, with $\hat{z}_0 = z_0 = h_\theta(s_0)$.
- A reward prediction loss, $\mathcal{L}_R = (R_\theta(\hat{z}_t, a_t) - r_{t+1})^2$.
- A forward dynamics loss (also called "latent state consistency loss"), $\mathcal{L}_f = \|f_\theta(\hat{z}_t, a_t) - h_{\theta^-}(s_{t+1})\|^2$, where θ^- are "target" parameters obtained by an exponential moving average of θ.
- A policy improvement loss, $\mathcal{L}_\pi = -Q_\theta(\hat{z}_t, \pi_\theta(\hat{z}_t))$, only optimized over the parameters of π_θ.

The first three losses are combined through a weighted sum, $\mathcal{L} = c_f \mathcal{L}_f + c_R \mathcal{L}_R + c_Q \mathcal{L}_Q$, which trains h_θ, f_θ, R_θ, and

[1] We generally do not call the intent embeddings "subgoals" in this paper because the Worker is not explicitly optimized to achieve them; instead, we are simply concatenating them to environment states.

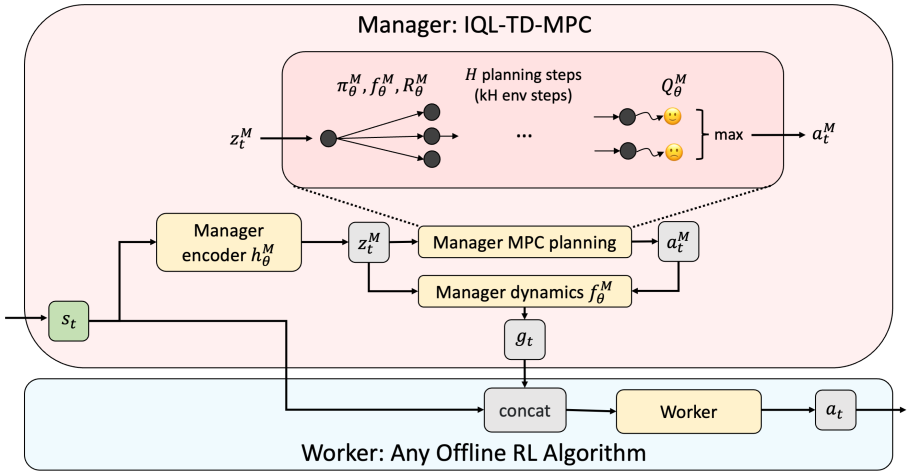

Figure 1. Overview of our hierarchical framework. The Manager is a model-based IQL-TD-MPC agent (inspired by Kostrikov et al. (2022) and Hansen et al. (2022)) that operates on a coarse timescale to generate intent embeddings g_t. To do so, the Manager performs Model Predictive Control over H planning steps (which is kH environment steps), using a learned policy π_θ^M, dynamics model f_θ^M, reward function R_θ^M, and critic Q_θ^M. Each intent g_t is concatenated with the state s_t and given to the Worker to output actions a_t. This Worker can be any offline RL algorithm.

Q_θ. The policy π_θ is trained independently by minimizing $\mathcal{L}_\pi$ without propagating gradients through either h_θ or Q_θ.

TD-MPC alternates between model training and acting in the environment. At inference time, TD-MPC plans in the latent space with MPC, which proceeds in three steps: (1) From current state s_0, set the first latent $z_0 = h_\theta(s_0)$, then generate n_π action sequences by unrolling the policy π_θ through the forward model f_θ over T steps: $a_t \sim \pi_\theta(\cdot \mid z_t)$ and $z_{t+1} = f_\theta(z_t, a_t)$. (2) Find optimal action sequences using Model Predictive Path Integral (MPPI) (Williams et al., 2015), which iteratively refines the mean and standard deviation of a Gaussian with diagonal covariance, starting from the above n_π action sequences combined with n_r additional random sequences sampled from the current Gaussian. The quality of an action sequence is obtained by $\sum_{t=0}^{T-1} \gamma^t R_\theta(z_t, a_t) + \gamma^T Q_\theta(z_T, a_T)$, i.e., unrolling the forward dynamics model for T steps, using the reward predictor to estimate the sum of rewards, and then bootstrapping with the value estimate of the critic. (3) Randomly select one from the best n_e action sequences from the last iteration of the previous step with weight proportional to its quality, and execute its first action in the environment.

3. Offline Model-Based RL via Implicit Q-Learning (IQL) and TD-MPC

In this section, we present IQL-TD-MPC, a framework that extends TD-MPC to the offline RL setting via Implicit Q-Learning (IQL) (Kostrikov et al., 2022). As we show in our experiments, naively training a vanilla TD-MPC agent on offline data performs poorly, as the model may suffer from out-of-distribution generalization errors when the training set has limited coverage. For instance, the state-action value function Q_θ may "hallucinate" very good actions never seen in the training set, and then the policy π_θ would learn to predict these actions. This could steer MPC planning into areas of the latent representation very far from the training distribution, compounding the error further.

IQL addresses the challenge of out-of-distribution actions by combining two ideas:

- Approximating the optimal value functions Q^* and V^* with TD-learning using only actions from the training set $\mathcal{D}$. This is achieved using the following loss on V_θ:[2]

$$\mathcal{L}_{V,IQL} = \mathbb{E}_{(s,a)\sim\mathcal{D}}[L_2^\tau(Q_{\theta^-}(s,a) - V_\theta(s))], \quad (1)$$

  where L_2^τ is the asymmetric squared loss $L_2^\tau(u) = |\tau - \mathbb{1}_{u<0}|u^2$, and $\tau \in (0.5, 1)$ is a hyper-parameter controlling the "optimality" of the learned value functions. The state-action value function Q_θ is optimized through the standard one-step TD loss:

$$\mathcal{L}_{Q,IQL} = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}[(Q_\theta(s,a) - (r + \gamma V_\theta(s')))^2]. \quad (2)$$

- Learning a policy using Advantage Weighted Regression (Peng et al., 2019), minimizing the loss:

$$\mathcal{L}_{\pi,IQL} = -\mathbb{E}_{(s,a)\sim\mathcal{D}}[\exp(\beta A_\theta(s,a)) \log \pi_\theta(a \mid s)], \quad (3)$$

  with advantage $A_\theta(s,a) = Q_{\theta^-}(s,a) - V_\theta(s)$, and $\beta > 0$ an inverse temperature hyper-parameter. The gradient of $\mathcal{L}_{\pi,IQL}$ flows only through π_θ, *not* through A_θ.

[2] For convenience, when describing the original IQL algorithm, we re-use notations V_θ, Q_θ, and π_θ even though they take raw states s as input rather than latent states z.

IQL avoids the out-of-distribution actions problem by restricting the policy to mimic actions from the data, while still outperforming the behavior policy by upweighting the best actions under Q_θ and V_θ.

To integrate IQL into TD-MPC (refer to Section 2.2), we first replace the TD-MPC policy improvement loss $\mathcal{L}_\pi$ with the IQL policy loss $\mathcal{L}_{\pi,IQL}$ (Eq. 3). This necessitates training an additional component not present in TD-MPC: a state value function V_θ to optimize $\mathcal{L}_{V,IQL}$ (Eq. 1) and $\mathcal{L}_{Q,IQL}$ (Eq. 2). As in TD-MPC (and contrary to IQL), all models are applied on learned latent states z. The state-action value function Q_θ may be trained with either the TD-MPC critic loss $\mathcal{L}_Q$ or the IQL critic loss $\mathcal{L}_{Q,IQL}$; the difference is whether bootstrapping is done using Q_θ itself or using V_θ. Our experiments typically use $\mathcal{L}_Q$ as we found it to give better results in practice.

This is not enough, however, to fully solve the out-of-distribution actions problem. The MPC planning may still prefer actions that lead to high-return states under R_θ and Q_θ, and hence exploit these models' blind spots. We propose the following fix: skip the iterative MPPI refinement of actions during planning, instead keeping only the best n_e sequences of actions among the n_π policy samples. This is a special case of the TD-MPC planning algorithm discussed in Section 2.2 where the number of random action sequences n_r is set to zero.

However, this fix brings in another issue: in the original implementation of TD-MPC, actions are sampled from the policy π_θ by $a \sim \mathcal{N}(\mu_\theta(z), \sigma^2)$, where μ_θ is a learned mean and σ decays linearly towards a fixed hyper-parameter value. If σ is too low, then the policy is effectively deterministic and all n_π samples will be nearly identical, which is problematic in our case because we are using $n_r = 0$. If σ is too high, then we again run into the problem of out-of-distribution actions. To avoid having to carefully tune σ, we learn a stochastic policy that outputs both $\mu_\theta(z)$ and a state-dependent $\sigma_\theta(z)$.[3]

With the above changes (using IQL losses, using only samples from the policy for planning, and learning a stochastic policy), IQL-TD-MPC preserves TD-MPC's ability to plan efficiently in a learned latent space, while benefiting from IQL's robustness to distribution shift in the offline setting.

4. IQL-TD-MPC as a Hierarchical Planner

We now turn to our second contribution, which is a hierarchical framework (Fig. 1) that uses IQL-TD-MPC as a Manager with *any* off-the-shelf offline RL algorithm as a Worker. This hierarchy aims to endow the agent with the ability to reason at longer time horizons. Indeed, although TD-MPC uses MPC planning to select actions, its planning horizon is typically short: Hansen et al. (2022) use a horizon of 5, and found no benefit from increasing it further due to compounding model errors. For sparse-reward tasks, this makes the planner highly dependent on the quality of the bootstrap estimates predicted by the critic Q_θ, which may be challenging to get right under complex dynamics.

We address this challenge by making IQL-TD-MPC operate as a Manager at a coarser timescale (adding the superscript M), processing trajectories $(s_0, a_0^M, r_k^M, s_k, a_k^M, \dots, r_{kH}^M, s_{kH})$ where:

- k is a hyper-parameter controlling the coarseness of the latent timescale, such that each latent transition skips over k low-level environment steps.
- H is the planning horizon; therefore, the effective environment-level horizon is kH.
- $r_{tk}^M = \sum_{i=(t-1)k+1}^{tk} r_i$, that is, Manager rewards sum up over the previous k environment steps.
- a_{tk}^M is an abstract action "summarizing" the transition from s_{tk} to $s_{(t+1)k}$.

How should these abstract actions be defined (Pertsch et al., 2021; Rosete-Beas et al., 2023)? Prior work learned an autoencoder that can reconstruct the next latent state (Mandlekar et al., 2020; Li et al., 2022), and one could define the abstract action as the latent representation of such an autoencoder. We adopt a similar approach in spirit, but tailored to our TD-MPC setup. Specifically, we train an "inverse dynamics" model b_θ^M in the latent space (instead of the raw environment state space):

$$a_{tk}^M = b_\theta^M(z_{tk}^M, z_{(t+1)k}^M), \tag{4}$$

where $z_i^M = h_\theta^M(s_i)$ is the Manager encoding of state s_i. b_θ^M is trained implicitly by backpropagating through a_t the gradient of the total loss. Similar to Director (Hafner et al., 2022), we found discrete actions to help, and thus modify the policy π_θ^M to output discrete actions (see Appendix A for details).

Once trained, the IQL-TD-MPC Manager can be used to generate "intent embeddings" to augment the state representation of any Worker that acts in the environment. We define the intent embedding $g_t \in \mathbb{R}^d$ at time t as the difference between the predicted next latent state and the current latent state:

$$g_t = f_\theta^M(z_t^M, a_t^M) - z_t^M, \tag{5}$$

where when training the Worker, a_t^M comes from the inverse dynamics model: $a_t^M = b_\theta^M(z_t^M, z_{t+k}^M)$. Note that we apply the intent embedding on *each* environment step, so Eq. 5 does *not* index by k.

The Worker can be any policy π. Its states are concatenated with the intent embeddings: $a_t \sim \pi(\cdot \mid \textsc{concat}(s_t, g_t))$.

[3] Our policy implementation is based on the Soft Actor-Critic (SAC) code from Yarats and Kostrikov (2020).

Since intent embeddings are in the Manager's latent space, the Manager may be trained independently from the Worker. In practice, we pre-train a single Manager for a task and use it with a range of different Workers (see Section 5.2 for experiments). A benefit of this concatenation strategy is its simplicity: it does not require modifying Worker training algorithms or losses, only appending intent embeddings to states during (i) offline dataset loading and (ii) evaluation.

4.1. Why are intent embeddings beneficial for offline RL?

Before turning to experiments, we provide an intuitive explanation for why we believe augmenting states with intent embeddings can be beneficial. For simplicity, we focus on the well-understood BC algorithm, but we note that many other offline RL algorithms such as AWAC, IQL, and TD3-BC use the BC objective in some way, and thus the intuition may also carry over to these algorithms.

We first provide an information-theoretic argument to explain why intent embedding should make the imitation learning objective easier to optimize. One of the primary obstacles in long-horizon sparse-reward offline RL is the ambiguity surrounding the relationship between each state-action pair in a dataset and its corresponding long-term objective. By incorporating intent embeddings derived from MPC planning into state-action pairs, our framework provides offline RL algorithms with a more well-defined association between each state-action pair and the objective being targeted. For a BC policy $\pi : \mathcal{S} \mapsto \mathcal{A}$, we typically train π to match the state-action pairs in the offline dataset. With intent embeddings, the agent can instead learn $\pi' : \mathcal{S} \times \mathbb{R}^d \mapsto \mathcal{A}$, which maps a pair of state and intent random variables (S_t, G_t) to an action random variable A_t. Since A_t is not independent of G_t given S_t, the mutual information $I((S_t, G_t); A_t) \geq I(S_t; A_t)$, so (S_t, G_t) contains at least as much information about A_t as S_t does on its own when learning a BC policy via imitation learning.

The above argument explains why it should be easier to optimize the BC objective when training the Worker, thanks to the additional information contained in the intent embedding. This can be particularly beneficial on offline datasets built from a mixture of varied policies (Fu et al., 2020). In addition to simplifying the task of the BC Worker, the Manager is trained to provide "good" intent embeddings at inference time. This is achieved through the MPC-based planning procedure of IQL-TD-MPC, by identifying a sequence of abstract actions $(a_t^M, a_{t+k}^M, \ldots, a_{t+kH}^M)$ that leads to high expected return (according to R_θ^M and Q_θ^M when unrolling f_θ^M). The intent embedding g_t, obtained from a_t^M through Eq. 5, is then used to condition the Worker policy, similar to prior work on goal-conditioned imitation learning (Mandlekar et al., 2020; Lynch et al., 2020).

5. Experiments

Our experiments aim to answer three questions: **(Q1)** How does IQL-TD-MPC perform as an offline RL algorithm, compared to both the original TD-MPC algorithm and other offline RL algorithms? **(Q2)** How much benefit do we obtain by using IQL-TD-MPC as a Manager in a hierarchical setting? **(Q3)** To what extent are the observed benefits actually coming from our IQL-TD-MPC algorithm?

Experimental Setup. We focus on continuous control tasks of the D4RL benchmark (Fu et al., 2020), following the experimental protocol from CORL (Tarasov et al., 2022): training with a batch size of 256 and reporting the normalized score (0 is random, 100 is expert) at the end of training, averaged over 100 evaluation episodes. Averages and standard deviations are reported over 5 random seeds. Each experiment was run on an A100 GPU. Training for a single seed (including both pre-training the Manager and training the Worker) took ~ 5 hours on average. See Appendix C for all hyper-parameters.

5.1. (Q1) How does IQL-TD-MPC perform as an offline RL algorithm?

We begin with a preliminary experiment to verify that IQL-TD-MPC is a viable offline RL algorithm. For this experiment, we compare IQL-TD-MPC, TD-MPC, and several offline RL algorithms from the literature on various tasks. See Table 1 for results. There are several key trends we can observe. First, vanilla TD-MPC does not perform well in general, and completely fails in the more difficult variants of the antmaze task. This is expected because TD-MPC is not designed to train from offline data. The one exception is the umaze environment in maze2d, where TD-MPC actually outperforms IQL-TD-MPC by a significant margin. We hypothesize that this is because the dynamics of this environment are very simple, and the data provides adequate coverage to learn effective TD-MPC models, while the conservative expectile updates of IQL-TD-MPC cause learning to be slower. The other trend we see is that IQL-TD-MPC is generally on par with the other offline RL algorithms. This confirms our hypothesis that IQL-TD-MPC is a viable model-based offline RL algorithm.

5.2. (Q2) How much benefit do we obtain by using IQL-TD-MPC as a Manager?

Now, we turn to the main results of our work, where we demonstrate the benefits of using IQL-TD-MPC as a Manager with a range of different non-hierarchical offline RL algorithms as Workers. For this experiment, we used offline RL algorithms from the CORL repository as Workers. We concatenated intent embeddings output by the Manager to the environment states seen by these Workers during

Table 1. Normalized scores of IQL-TD-MPC, other offline RL algorithms (IQL (Kostrikov et al., 2022), TT (Janner et al., 2021), TAP (Jiang et al., 2022)) and TD-MPC on D4RL after 1M training steps. IQL results are from Tarasov et al. (2022). TT and TAP results are from their papers, except for antmaze-umaze-diverse and maze2d, which we reproduced with the default hyperparameters because they were not reported. Each entry shows the mean over 100 episodes and 5 seeds, and the standard deviation over seeds. Bolded numbers are within one standard deviation of the best result in each row.

Dataset \ Algorithm	IQL	TT	TAP	TD-MPC	IQL-TD-MPC
antmaze-umaze-v2	87.5 ± 2.6	$\mathbf{100.0} \pm 0.0$	81.5 ± 2.8	44.6 ± 28.2	52.0 ± 46.0
antmaze-umaze-diverse-v2	$\mathbf{66.2} \pm 13.8$	21.5 ± 2.9	$\mathbf{68.5} \pm 3.3$	0.0 ± 0.0	$\mathbf{72.6} \pm 26.6$
antmaze-medium-play-v2	71.5 ± 12.6	$\mathbf{93.3} \pm 6.4$	78.0 ± 4.4	1.8 ± 3.91	$\mathbf{88.8} \pm 5.9$
antmaze-medium-diverse-v2	70.0 ± 10.9	$\mathbf{100.0} \pm 0.0$	85.0 ± 3.6	0.0 ± 0.0	40.3 ± 34.2
antmaze-large-play-v2	40.8 ± 12.7	66.7 ± 12.2	$\mathbf{74.0} \pm 4.4$	0.0 ± 0.0	66.6 ± 13.7
antmaze-large-diverse-v2	47.5 ± 9.5	60.0 ± 12.7	$\mathbf{82.0} \pm 5.0$	0.0 ± 0.0	4.0 ± 4.1
antmaze-ultra-play-v0	9.2 ± 6.7	$\mathbf{20.0} \pm 10.0$	$\mathbf{22.0} \pm 4.1$	0.0 ± 0.0	$\mathbf{20.6} \pm 16.0$
antmaze-ultra-diverse-v0	$\mathbf{22.5} \pm 8.3$	$\mathbf{33.3} \pm 12.2$	$\mathbf{26.0} \pm 4.4$	0.0 ± 0.0	3.6 ± 10.1
maze2d-umaze-v1	37.7 ± 2.0	36.7 ± 2.1	$\mathbf{58.6} \pm 1.4$	$\mathbf{76.4} \pm 20.8$	40.9 ± 45.3
maze2d-medium-v1	35.5 ± 1.0	32.7 ± 1.1	-3.9 ± 0.3	85.3 ± 15.8	$\mathbf{161.0} \pm 11.3$
maze2d-large-v1	49.6 ± 22.0	33.2 ± 1.0	-2.1 ± 0.1	$\mathbf{121.6} \pm 27.0$	$\mathbf{158.9} \pm 77.1$
halfcheetah-medium-v2	48.3 ± 0.11	46.9 ± 0.4	45.0 ± 0.1	45.7 ± 14.6	$\mathbf{57.4} \pm 0.1$
halfcheetah-medium-replay-v2	44.2 ± 1.2	41.9 ± 2.5	40.8 ± 0.6	45.7 ± 5.0	$\mathbf{49.2} \pm 1.3$
halfcheetah-medium-expert-v2	94.6 ± 0.2	$\mathbf{95.0} \pm 0.2$	91.8 ± 0.8	-1.0 ± 0.9	44.8 ± 8.5

both training and evaluation. Once the boilerplate code was written, the changes to the CORL algorithms were straightforward, since they typically only required adding two lines of code to (i) augment states in the offline dataset and (ii) wrap the evaluation environment.

Table 2 shows the results of this experiment, for the following CORL Workers: AWAC, BC, DT, IQL, TD3-BC, and CQL. Overall, we observe a dramatic improvement in performance for all these agents compared to their baseline versions, whose only difference is the lack of intent embeddings concatenated to state vectors. Interestingly, vanilla AWAC / BC / DT / TD3-BC all get a zero score on the large and ultra variants of the antmaze task, while with our modification, they are able to learn to solve the task. This shows that the intent embeddings produced by the Manager are highly useful, and can be used to compensate for the lack of long-term planning abilities in off-the-shelf RL agents.

Notably, our approach slightly worsens performance on the half-cheetah locomotion tasks. A likely explanation is that these tasks are more about fine-grained control and thus have less natural hierarchical structure for our framework to exploit. The intent embeddings are trained by having the Manager look at states k timesteps ahead, but lookahead may not help but even hurt on these tasks as it restricts the pool of candidate actions the Worker is considering.

In Fig. 2, we visualize an episode of the BC agent on the antmaze-large-play-v2 task, in order to qualitatively understand the benefits of our framework. On the left, we see that without intent embeddings, the ant gets stuck close to the start of the maze, never reaching the goal. On the right, we see that the ant reaches the goal, guided by the intent embeddings visualized in green. To generate these green visualizations, we trained a separate decoder alongside the IQL-TD-MPC Manager that converts the intent embeddings (in the Manager's latent space) back into the raw environment state space, which contains the position and velocity of the ant. The green dot shows the position, and the green line attached to the dot shows the velocity (speed is the length of the line). This decoder was trained on a reconstruction loss and did not affect the training of the other models. Overall, this visualization shows that the intent embeddings are effectively acting as latent-space subgoals that the Worker exploits to learn a more effective policy.

5.3. (Q3) To what extent are the observed benefits coming from IQL-TD-MPC?

One may wonder whether the strong results in Table 2 are simply due to a "regularization" effect, or whether the intent embeddings simply tie-break the stochasticity of the behavior policy. We conduct an ablation to address this: we run our framework, but replace the intent embeddings with random vectors of the same dimensionality, with entries drawn uniformly from $(0, 1)$. See Table 3 for results.

Across nearly all tasks and algorithms, we found no statistically significant difference between the baseline and the ablation. This means that the Workers typically learned to ignore the random vectors. Comparing against the clear benefits of our proposed method in Table 2, we can conclude

Table 2. Results of our hierarchical framework, where we append IQL-TD-MPC Manager intents to states in various offline RL algorithms taken from the CORL repository (Tarasov et al., 2022). Each table entry is of the form "baseline evaluation score $\rightarrow$ our evaluation score". In either case, we report scores after 500K steps of training; in general, we found that all agents plateaued after this point. For our hierarchical framework, these 500K steps correspond to 300K steps of pre-training the Manager, followed by 200K steps of training the CORL Worker. All entries report a mean over 5 independent random seeds; see Table 4 in Appendix B for standard deviations. Green entries indicate statistically significant improvement, while red entries indicate statistically significant degradation.

Dataset \ Algorithm	AWAC	BC	DT	IQL	TD3-BC	CQL
antmaze-umaze-v2	$51 \rightarrow 86$	$52 \rightarrow 78$	$64 \rightarrow 89$	$44 \rightarrow 80$	$90 \rightarrow 82$	$67 \rightarrow 69$
antmaze-umaze-diverse-v2	$53 \rightarrow 60$	$49 \rightarrow 48$	$55 \rightarrow 38$	$60 \rightarrow 51$	$45 \rightarrow 53$	$37 \rightarrow 36$
antmaze-medium-play-v2	$0 \rightarrow 36$	$0 \rightarrow 52$	$0 \rightarrow 43$	$70 \rightarrow 64$	$0.2 \rightarrow 60$	$0.8 \rightarrow 33$
antmaze-medium-diverse-v2	$0.8 \rightarrow 16$	$0.2 \rightarrow 20$	$0.2 \rightarrow 33$	$63 \rightarrow 30$	$0.4 \rightarrow 21$	$0.2 \rightarrow 14$
antmaze-large-play-v2	$0 \rightarrow 67$	$0 \rightarrow 50$	$0 \rightarrow 53$	$54 \rightarrow 70$	$0 \rightarrow 46$	$0 \rightarrow 19$
antmaze-large-diverse-v2	$0 \rightarrow 40$	$0 \rightarrow 38$	$0 \rightarrow 31$	$31 \rightarrow 46$	$0 \rightarrow 29$	$0 \rightarrow 16$
antmaze-ultra-play-v0	$0 \rightarrow 18$	$0 \rightarrow 18$	$0 \rightarrow 10$	$9 \rightarrow 16$	$0 \rightarrow 20$	$0 \rightarrow 5$
antmaze-ultra-diverse-v0	$0 \rightarrow 37$	$0 \rightarrow 35$	$0 \rightarrow 10$	$22 \rightarrow 27$	$0 \rightarrow 29$	$0.6 \rightarrow 5$
maze2d-umaze-v1	$77 \rightarrow 78$	$3 \rightarrow 64$	$26 \rightarrow 63$	$41 \rightarrow 77$	$39 \rightarrow 77$	$-14 \rightarrow 7$
maze2d-medium-v1	$43 \rightarrow 67$	$3 \rightarrow 70$	$13 \rightarrow 71$	$32 \rightarrow 78$	$101 \rightarrow 47$	$104 \rightarrow 16$
maze2d-large-v1	$193 \rightarrow 132$	$-1 \rightarrow 94$	$3 \rightarrow 96$	$42 \rightarrow 135$	$69 \rightarrow 126$	$53 \rightarrow 64$
halfcheetah-medium-v2	$49 \rightarrow 45$	$42 \rightarrow 45$	$42 \rightarrow 47$	$47 \rightarrow 43$	$47 \rightarrow 44$	$46 \rightarrow 44$
halfcheetah-medium-replay-v2	$45 \rightarrow 41$	$34 \rightarrow 40$	$39 \rightarrow 37$	$44 \rightarrow 40$	$44 \rightarrow 39$	$45 \rightarrow 32$
halfcheetah-medium-expert-v2	$95 \rightarrow 80$	$57 \rightarrow 84$	$63 \rightarrow 52$	$92 \rightarrow 79$	$86 \rightarrow 76$	$90 \rightarrow 45$

Table 3. Ablation results, where we replace Manager intent embeddings with random vectors. Each table entry is of the form "baseline evaluation score $\rightarrow$ ablation evaluation score". We report scores after 500K steps of training. All entries report a mean over 5 independent random seeds; see Table 5 in Appendix B for standard deviations. Red entries indicate statistically significant degradation.

Dataset \ Algorithm	AWAC	BC	IQL	TD3-BC
antmaze-medium-play-v2	$0 \rightarrow 0$	$0 \rightarrow 0$	$70 \rightarrow 66$	$0.2 \rightarrow 0$
antmaze-medium-diverse-v2	$0.8 \rightarrow 0.2$	$0.2 \rightarrow 0$	$63 \rightarrow 71$	$0.4 \rightarrow 0.2$
antmaze-large-play-v2	$0 \rightarrow 0$	$0 \rightarrow 0$	$54 \rightarrow 25$	$0 \rightarrow 0$
antmaze-large-diverse-v2	$0 \rightarrow 0$	$0 \rightarrow 0$	$31 \rightarrow 37$	$0 \rightarrow 0$
halfcheetah-medium-v2	$49 \rightarrow 49$	$42 \rightarrow 42$	$47 \rightarrow 47$	$47 \rightarrow 47$
halfcheetah-medium-replay-v2	$45 \rightarrow 43$	$34 \rightarrow 34$	$44 \rightarrow 43$	$44 \rightarrow 44$
halfcheetah-medium-expert-v2	$95 \rightarrow 93$	$57 \rightarrow 61$	$92 \rightarrow 90$	$86 \rightarrow 87$

that IQL-TD-MPC was critical; it guides the Workers in a more impactful way than just regularization.

Interestingly, Table 3 shows that the Workers have learned to ignore the random vectors in half-cheetah (performance is unchanged), while in Table 2, our modification *harmed* performance. This confirms that the intent embeddings are correlated with environment states in a way that RL algorithms do not ignore, which may help or hurt depending on how much hierarchical structure the task has.

6. Related Work

6.1. Offline Reinforcement Learning

In offline reinforcement learning (Levine et al., 2020; Prudencio et al., 2023; Lange et al., 2012; Ernst et al., 2005), the agent learns from a fixed offline dataset. Li et al. (2022) learn a generative model of potential goals to pursue given the current state, along with a goal-conditioned policy trained by CQL, from a combination of the task reward with a goal-reaching reward. Planning is performed by optimizing goals (with CEM) to maximize those rewards as estimated by the value function of the policy over the planning horizon. In our work, by contrast, our intent embeddings are defined in the Manager's learned latent space, and we can use this Manager with any offline RL Worker. The recently proposed POR algorithm (Xu et al., 2022) learns separate "guide" and "execute" policies, where the "guide" policy abstracts out the action space. Our Manager can also be seen as such a guide that would plan over longer time horizons.

A recent line of work uses Transformers (Vaswani et al., 2017) to model the trajectories in the offline dataset (Janner et al., 2021; Chen et al., 2021). Jiang et al. (2022) propose the Trajectory Autoencoding Planner (TAP), that models a trajectory by a sequence of discrete tokens learned by a Vector Quantised-Variational AutoEncoder (VQ-VAE, van den Oord et al., 2017), conditioned on the initial state. This enables efficient search with a Transformer-based trajectory generative model. One can relate this approach to ours

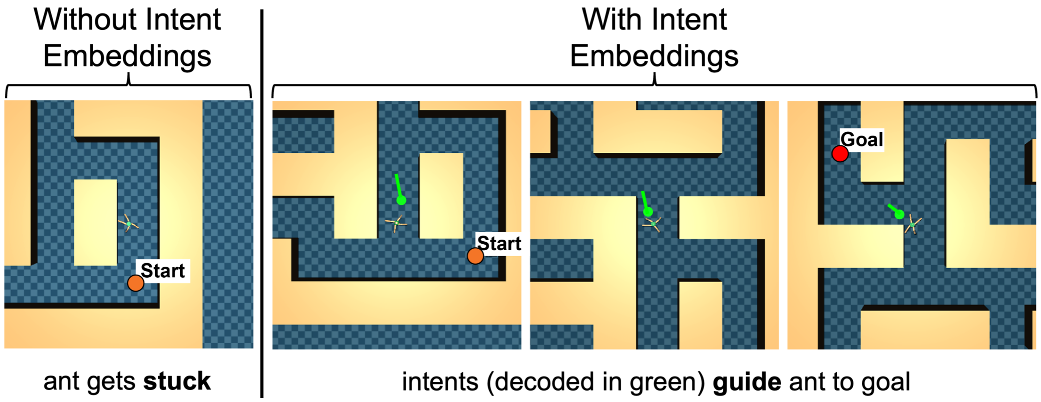

Figure 2. Visualization of an episode of the Behavioral Cloning (BC) agent on the antmaze-large-play-v2 task. On the left, without intent embeddings, the ant gets stuck close to the start of the maze, never reaching the goal. On the right, the ant reaches the goal, guided by the intent embeddings whose decoding is visualized in green. We see that the intent embeddings act as latent-space subgoals.

by interpreting the generation of encoded trajectories as the Manager, and the decoding into actual actions as the Worker. However, this distinction is somewhat artificial since in contrast to our approach, the Manager provides an intent embedding that encodes an entire predicted trajectory, rather than a single state. In addition, TAP relies on a Monte-Carlo "return-to-go" estimator to bootstrap search, while we explicitly learn a temporally abstract Manager value function.

Play-LMP (Lynch et al., 2020) encodes goal-conditioned sub-trajectories in a latent space through a conditional sequence-to-sequence VAE (Sohn et al., 2015), which can be used to sample latent plans that are decoded through a goal-conditioned policy. However, there is no notion of optimizing a task reward here: instead, the desired goal state must be provided as input to the model to solve a task.

6.2. Hierarchical Reinforcement Learning

Though our work focuses on the offline setting, we highlight a few related works in the online setting. Director (Hafner et al., 2022) trains Manager and Worker policies in imagination, where the Manager actions are discrete representations of goals for the Worker, learned from the latent representation of a world model. Although we re-use a similar discrete representation for Manager actions, this approach differs from our work in several ways: it focuses on the online setting, there is no planning during inference, and the world model is not temporally abstract. Our work may be related to the literature on option discovery (Sutton et al., 1999; Bagaria and Konidaris, 2020; Daniel et al., 2016; Brunskill and Li, 2014). In our proposed hierarchical framework, the intent embeddings output by our Manager can be seen as latent skills (Pertsch et al., 2021; Rosete-Beas et al., 2023) that the Worker conditions on to improve its learning efficiency. Finally, our work can be seen as an instantiation of one piece of the H-JEPA framework laid out by LeCun (2022): we learn a Manager world model at a higher level of temporal abstraction, which works in tandem with a Worker to optimize rewards.

7. Limitations and Future Work

In this paper, we propose a non-trivial extension of TD-MPC to the offline setting based on IQL, and leverage its superior planning abilities as a temporally extended Manager in a hierarchical architecture. Our experiments confirm the benefits of this hierarchical framework in guiding offline RL algorithms.

Our algorithm still suffers from a number of limitations that we intend to tackle in future work: (1) Our method hurts performance on some locomotion tasks (Table 2), which require fine-grained control. It is unsurprising that hierarchy does not help in such contexts; however, further investigation is required to confirm our intuition for why the Worker algorithms are unable to simply ignore these harmful intent embeddings. (2) The Worker agent may also be improved by actively planning toward the intent embedding set by the Manager. For instance, the Worker itself could be an IQL-TD-MPC agent modeling the world at the original environment timescale, unlike the temporally abstract Manager. (3) Our Manager's timescale is defined by a fixed hyperparameter k. This could instead be set dynamically by the Manager, and included in the intent embedding concatenated to the environment state. (4) Similar to the TD-MPC algorithm we build on, our approach is computationally intensive, both during Manager pre-training and inference, because we need to unroll the Manager's world model to obtain the intent embeddings. A potential avenue to speed it up could be to represent the world model as a Transformer (Vaswani et al., 2017; Micheli et al., 2022), for more efficient rollouts.

References

Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning, 2021.

Akhil Bagaria and George Konidaris. Option discovery using deep skill chaining. In *International Conference on Learning Representations*, 2020.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013.

Matthew Botvinick and Ari Weinstein. Model-based hierarchical reinforcement learning and human action control. *Philos Trans R Soc Lond B Biol Sci*, 369(1655), November 2014.

Emma Brunskill and Lihong Li. Pac-inspired option discovery in lifelong reinforcement learning. In *International conference on machine learning*, pages 316–324. PMLR, 2014.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

Ignasi Clavera, Violet Fu, and Pieter Abbeel. Model-augmented actor-critic: Backpropagating through paths, 2020.

Christian Daniel, Herke Van Hoof, Jan Peters, and Gerhard Neumann. Probabilistic inference for determining options in reinforcement learning. *Machine Learning*, 104:337–357, 2016.

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

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

Scott Fujimoto and Shixiang (Shane) Gu. A minimalist approach to offline reinforcement learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 20132–20145. Curran Associates, Inc., 2021.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NeurIPS'18, pages 2455–2467, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, pages 2555–2565, 2019.

Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels, 2022. URL https://arxiv.org/abs/2206.04114.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*, 2021.

Zhengyao Jiang, Tianjun Zhang, Michael Janner, Yueying Li, Tim Rocktäschel, Edward Grefenstette, and Yuandong Tian. Efficient planning in a compact latent action space, 2022. URL https://arxiv.org/abs/2208.10291.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with Implicit Q-Learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1179–1191. Curran Associates, Inc., 2020.

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

Yann LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62, 2022.

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

Jinning Li, Chen Tang, Masayoshi Tomizuka, and Wei Zhan. Hierarchical planning through goal-conditioned offline reinforcement learning. *IEEE Robotics and Automation Letters*, 7(4):10216–10223, 2022. doi: 10.1109/LRA.2022.3190100.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 1113–1132. PMLR, 30 Oct–01 Nov 2020.

Ajay Mandlekar, Fabio Ramos, Byron Boots, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Dieter Fox. IRIS: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data, 2020. URL https://arxiv.org/abs/1911.05321.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample efficient world models, 2022. URL https://arxiv.org/abs/2209.00588.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets, 2020. URL https://arxiv.org/abs/2006.09359.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning, 2019. URL https://arxiv.org/abs/1910.00177.

Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*, pages 188–204. PMLR, 2021.

Rafael Figueiredo Prudencio, Marcos ROA Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Martin L Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

Erick Rosete-Beas, Oier Mees, Gabriel Kalweit, Joschka Boedecker, and Wolfram Burgard. Latent plans for task-agnostic offline reinforcement learning. In *Conference on Robot Learning*, pages 1838–1849. PMLR, 2023.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, Go, Chess and Shogi by planning with a learned model. *Nature*, 588(7839):604–609, Dec 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-03051-4. URL https://doi.org/10.1038/s41586-020-03051-4.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Denis Tarasov, Alexander Nikulin, Dmitry Akimov, Vladislav Kurenkov, and Sergey Kolesnikov. CORL: Research-oriented deep offline reinforcement learning library. In *3rd Offline RL Workshop: Offline RL as a "Launchpad"*, 2022. URL https://openreview.net/forum?id=SyAS49bBcv.

Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. Neural discrete representation learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. Model predictive path integral control using covariance variable importance sampling. *CoRR*, abs/1509.01149, 2015. URL http://arxiv.org/abs/1509.01149.

Haoran Xu, Li Jiang, Jianxiong Li, and Xianyuan Zhan. A policy-guided imitation approach for offline reinforcement learning, 2022. URL https://arxiv.org/abs/2210.08323.

Denis Yarats and Ilya Kostrikov. Soft actor-critic (sac) implementation in pytorch. https://github.com/denisyarats/pytorch_sac, 2020.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing*

Systems, volume 34, pages 25476–25488. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/d5eca8dc3820cad9fe56a3bafda65ca1-Paper.pdf.

Appendix

A. Discrete Manager Actions in IQL-TD-MPC

Similar to Hafner et al. (2022), we define the Manager's action as a vector of several categorical variables. The inverse dynamics model b_θ^M (Eq. 4) takes latent states z_{tk}^M and $z_{(t+1)k}^M$ as input and outputs a matrix of $L \times C$ logits, representing L categorical distributions with C categories each. The model then samples a C-dimensional one-hot vector from each of the L distributions, and flattens the results into a sparse binary vector of length $L \times C$. See Figure G.1 from Hafner et al. (2022) for a visualization. This sparse binary vector serves as the "action" chosen by the Manager. The model is optimized end-to-end together with all other components of IQL-TD-MPC using straight-through gradients (Bengio et al., 2013). Unlike Hafner et al. (2022), we did not include a KL-divergence regularization term in the model objective as we found regularizing the distribution towards some uniform prior hurts the final performance. In all our experiments, we used $L = 8$ and $C = 10$.

As a result of this change, we must also change the Manager policy network π_θ^M to output discrete actions. Hence, we modify π_θ^M to output L categorical distributions of size C (applying a softmax instead of the squashed Normal distribution used for continuous actions in IQL-TD-MPC). The behavioral cloning term $\log \pi_\theta(a \mid s)$ in the IQL policy loss (Eq. 3) thus becomes a cross-entropy loss over the C categories. This loss is summed over the L categorical distributions, which are treated independently.

B. Standard Deviation Tables

We provide standard deviations accompanying the results in the main text. Table 4 provides standard deviations for Table 2, and Table 5 provides standard deviations for Table 3.

Table 4. Standard deviations accompanying the means reported in Table 2. The table is formatted in the same way, so all these standard deviations are in the same positions as their corresponding means.

Dataset \ Algorithm	AWAC	BC	DT	IQL	TD3-BC	CQL
antmaze-umaze-v2	$9 \to 12$	$6 \to 2$	$4 \to 3$	$4 \to 4$	$3 \to 3$	$7 \to 3$
antmaze-umaze-diverse-v2	$10 \to 8$	$5 \to 7$	$6 \to 11$	$7 \to 18$	$6 \to 3$	$21 \to 3$
antmaze-medium-play-v2	$0 \to 13$	$0 \to 8$	$0 \to 11$	$5 \to 6$	$0.4 \to 4$	$0.8 \to 10$
antmaze-medium-diverse-v2	$1 \to 9$	$0.4 \to 8$	$0.5 \to 11$	$6 \to 7$	$0.5 \to 1$	$0.4 \to 6$
antmaze-large-play-v2	$0 \to 10$	$0 \to 3$	$0 \to 3$	$9 \to 7$	$0 \to 8$	$0 \to 5$
antmaze-large-diverse-v2	$0 \to 5$	$0 \to 5$	$0 \to 7$	$11 \to 9$	$0 \to 1$	$0 \to 5$
antmaze-ultra-play-v0	$0 \to 8$	$0 \to 4$	$0 \to 2$	$6 \to 8$	$0 \to 6$	$0 \to 2$
antmaze-ultra-diverse-v0	$0 \to 12$	$0 \to 8$	$0 \to 3$	$8 \to 18$	$0 \to 7$	$0.8 \to 2$
maze2d-umaze-v1	$38 \to 5$	$4 \to 9$	$12 \to 1$	$1 \to 3$	$14 \to 2$	$0.8 \to 42$
maze2d-medium-v1	$21 \to 21$	$5 \to 7$	$3 \to 7$	$7 \to 10$	$49 \to 8$	$15 \to 42$
maze2d-large-v1	$20 \to 35$	$0.5 \to 8$	$2 \to 8$	$21 \to 30$	$21 \to 66$	$61 \to 90$
halfcheetah-medium-v2	$0.2 \to 0.1$	$0.2 \to 0.2$	$0.3 \to 0.5$	$0.3 \to 0.4$	$0.2 \to 0.3$	$0.1 \to 0.1$
halfcheetah-medium-replay-v2	$0.1 \to 0.3$	$0.7 \to 0.7$	$0.2 \to 1$	$0.3 \to 1$	$0.2 \to 1$	$0.4 \to 5$
halfcheetah-medium-expert-v2	$0.8 \to 9$	$6 \to 4$	$7 \to 5$	$0.9 \to 7$	$8 \to 5$	$2 \to 2$

Table 5. Standard deviations accompanying the means reported in Table 3. The table is formatted in the same way, so all these standard deviations are in the same positions as their corresponding means.

Dataset \ Algorithm	AWAC	BC	IQL	TD3-BC
antmaze-medium-play-v2	$0 \to 0$	$0 \to 0$	$5 \to 2$	$0.4 \to 0$
antmaze-medium-diverse-v2	$1 \to 0.4$	$0.4 \to 0$	$6 \to 7$	$0.5 \to 0.4$
antmaze-large-play-v2	$0 \to 0$	$0 \to 0$	$9 \to 6$	$0 \to 0$
antmaze-large-diverse-v2	$0 \to 0$	$0 \to 0$	$11 \to 12$	$0 \to 0$
halfcheetah-medium-v2	$0.2 \to 0.3$	$0.2 \to 0.2$	$0.3 \to 0.1$	$0.2 \to 0.4$
halfcheetah-medium-replay-v2	$0.1 \to 0.4$	$0.7 \to 1$	$0.3 \to 0.7$	$0.2 \to 0.4$
halfcheetah-medium-expert-v2	$0.8 \to 2$	$6 \to 5$	$0.9 \to 2$	$8 \to 5$

C. Hyper-parameters

In this section, we list all hyper-parameters used in experiments. Table 6 contains hyper-parameters that were already present in the original TD-MPC algorithm (or that we added to slightly tweak its behavior, e.g., the ability to disable Prioritized Experience Replay or to use the policy mean in the TD target instead of a sample). Changes compared to the original TD-MPC implementation (`https://github.com/nicklashansen/tdmpc`) are bolded.

Table 7 lists the hyper-parameters for IQL-TD-MPC, related to integrating the IQL losses and making the continuous policy π_θ stochastic. Table 8 lists the hyper-parameters for using IQL-TD-MPC as a Manager (using a discrete stochastic policy π_θ, as discussed in Appendix A).

Table 6. TD-MPC hyper-parameters that we use in our IQL-TD-MPC algorithm. Bolded values are those that were modified compared to the original TD-MPC implementation from Hansen et al. (2022). We found no benefit to increasing the planning horizon beyond 2 in the offline setting. The motivation for changing n_π and n_r is described in Section 3. We disabled Prioritized Experience Replay out of caution in the offline setting, to be sure that the initial arbitrary priority (assigned to all transitions in the buffer after loading the dataset) would not artificially bias the sampling distribution (a problem that does not occur in the online setting, where each new transition gets assigned the maximum priority seen so far). Decreasing the learning rate and using the policy mean for bootstrapping were found to lead to more stable results for some tasks. Using a smaller batch size was purely for the purpose of fair comparison with prior results reported in the literature.

hyper-parameter	Value in TD-MPC	Value in IQL-TD-MPC
γ	0.99	0.99
latent dimension	50	50
planning horizon H	**5**	**2**
CEM population size	512	512
CEM #policy actions (n_π)	**25**	**512**
CEM #random actions (n_r)	**487**	**0**
CEM elite size (n_e)	64	64
CEM iterations	6	6
CEM momentum coefficient	0.1	0.1
CEM temperature	0.5	0.5
enable Prioritized Experience Replay	**yes**	**no**
learning rate	$\mathbf{10^{-3}}$	$\mathbf{3 \cdot 10^{-4}}$
batch size	**512**	**256**
MLP hidden size	512	512
encoder / decoder hidden size	256	256
bootstrapping value on last planning state	$\boldsymbol{Q_\theta(s, \pi_\theta(s_H))}$	$\boldsymbol{Q_\theta(s, \mathbb{E}_{a \sim \pi_\theta(a\|s_H)}[a])}$
bootstrapping value in TD target	$\boldsymbol{Q_\theta(s', \pi_\theta(s'))}$	$\boldsymbol{Q_\theta(s', \mathbb{E}_{a' \sim \pi_\theta(a'\|s')}[a'])}$
reward loss coefficient (c_R)	0.5	0.5
critic loss coefficient (c_Q)	0.1	0.1
consistency loss coefficient (c_f)	2	2
temporal coefficient (ρ)	0.5	0.5
gradient clipping threshold	10	10
θ^- update frequency	2	2
θ^- update momentum	0.01	0.01

Table 7. Hyper-parameters that we introduced specifically for our IQL-TD-MPC algorithm in the "flat" (non-hierarchical) setting described in Section 3. These hyper-parameters were used to obtain the results in Table 1. The action clipping threshold clips actions from the offline dataset to avoid infinite loss $\mathcal{L}_{\pi,IQL}$ (Eq. 3). The entropy bonus weight is the coefficient of an extra term we add to $\mathcal{L}_{\pi,IQL}$ to maximize entropy so as to prevent policy collapse. This term is approximated as $\log \pi_\theta(s)$, where $\pi_\theta(s)$ is a random action sampled from $\pi_\theta(\cdot \mid s)$.

hyper-parameter	**Value in IQL-TD-MPC**
IQL τ	0.9
IQL β	3
exponential advantage threshold	100
loss for critic Q_θ	$\mathcal{L}_Q$ (exception: $\mathcal{L}_{Q,IQL}$ for antmaze-{medium,large,ultra}-* tasks)
critic loss $\mathcal{L}_{V,IQL}$ coefficient	0.1
stochastic policy $\log \sigma$ (std) range	$(-5, 2)$
stochastic policy action clipping threshold	0.99
stochastic policy entropy bonus weight	0.1

Table 8. Hyper-parameters that were used specifically for our IQL-TD-MPC algorithm in the setting described in Section 4 where IQL-TD-MPC is a Manager. These hyper-parameters were used to obtain the results in Table 2 and Table 3. Compared to "flat" IQL-TD-MPC (Table 6 and Table 7), we decreased the latent dimension as we found no benefit in using higher values, while increasing the planning horizon for the Manager proved useful. The reward scale factor scales down manager rewards in tasks where otherwise summing rewards over k timesteps can lead to high Q-values and an explosion of critic losses. The IQL inverse temperature β is also updated accordingly to "cancel out" the effect of this rescaling in the advantage weight computation.

hyper-parameter	**Value in IQL-TD-MPC when used as a Manager**
latent dimension	10
planning horizon H	4
reward scale factor	0.1 for maze2d and locomotion tasks, 1.0 for antmaze tasks
IQL τ	0.9
IQL β	$3/$reward scale factor
exponential advantage threshold	100
loss for critic Q_θ	$\mathcal{L}_Q$ (exception: $\mathcal{L}_{Q,IQL}$ for antmaze-{medium,large,ultra}-* tasks)
critic loss $\mathcal{L}_{V,IQL}$ coefficient	0.1
latent timescale coarseness k	8
discrete policy L (Appendix A)	8
discrete policy C (Appendix A)	10