2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

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

Rohan Chitnis*, Yingchen Xu*, Bobak Hashemi, Lucas Lehnert, Urun Dogan, Zheqing Zhu, Olivier Delalleau†
Meta AI, FAIR

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); and 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 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.**

I. 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 [1], [2], [3], has shown great promise due to its sample efficiency compared to its model-free counterpart [4], [5]. 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 [6], [7]. 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 [8]. Modeling the world at a higher level of abstraction can enable predicting long-term future outcomes more accurately and efficiently.

Long-horizon sparse-reward tasks are particularly challenging for offline RL, since the agent cannot explore the environment [9], [10], [11], [12]. The offline setting is key to training RL agents safely, but poses unique challenges such as value mis-estimation [9]. In this paper, we study offline *model-based* RL, and hypothesize that planning in a learned

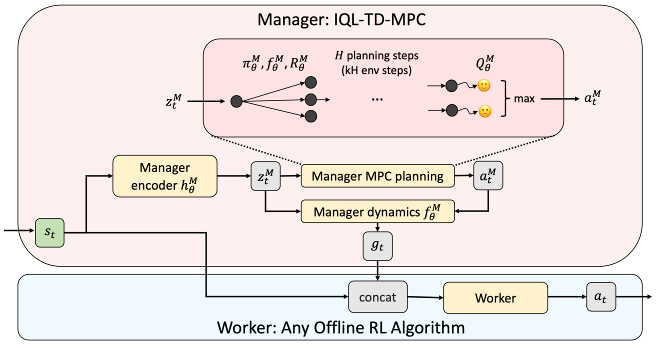

Fig. 1: Overview of our hierarchical framework. The Manager is a model-based IQL-TD-MPC agent (inspired by [13] and [14]) that operates on a coarse timescale. It generates intent embeddings g_t by performing Model Predictive Control over H planning steps (which is kH environment steps), using a learned 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.

temporally abstract model of the environment can produce significant improvements over "flat" algorithms that do not use temporal abstraction. Our two key contributions are:

- **Section IV**: 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) [13] with the popular offline RL algorithm Implicit Q-Learning (IQL) [14]. This combination requires several non-trivial design decisions.
- **Section V**: 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. Exper-

* Equal contribution
† Currently at NVIDIA, but work done while at Meta AI (FAIR team)

[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.

imentally, we study the D4RL benchmark [15], showing that IQL-TD-MPC is far superior to vanilla TD-MPC and on par with several other popular offline RL algorithms in most settings. Then, we show the significant benefits of our proposed hierarchical framework. For instance, the well-established offline RL algorithms AWAC [16], TD3-BC [17], DT [18], and CQL [19] 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`.

II. Related Work

A. Offline Reinforcement Learning

In offline RL [9], [10], [11], [12], the agent learns from a fixed offline dataset. [20] learns a generative model of potential goals to pursue given the current state, along with a goal-conditioned policy trained by Conservative Q-Learning (CQL [19]), 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 [21] 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 [22] to model trajectories from the offline dataset [23], [18]. [24] proposes the Trajectory Autoencoding Planner (TAP), that models a trajectory by a sequence of discrete tokens learned by a Vector Quantised-Variational AutoEncoder (VQ-VAE [25]), conditioned on the initial state. One can see the generation of encoded trajectories as a Manager, and the decoding into actual actions as a Worker. However, in contrast to our approach, this 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 [26] encodes goal-conditioned sub-trajectories in a latent space through a conditional VAE [27], 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.

B. Hierarchical Reinforcement Learning

Though our work focuses on the offline setting, we highlight a few related works in the online setting. Director [28] 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 [29], [30], [31], [32]. In our proposed hierarchical framework, the intent embeddings output by our Manager can be seen as latent skills [33], [34] 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 [35]: we learn a Manager world model at a higher level of temporal abstraction, which works in tandem with a Worker to optimize rewards.

III. Preliminaries

A. Markov Decision Processes and Offline RL

We consider the standard infinite-horizon Markov Decision Process (MDP) [36] 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 [9], [10], the agent learns from a fixed dataset rather than collecting its own data. 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 [37].

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

Our work builds on TD-MPC [13], 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 $\pi_\theta : \mathbb{R}^d \times \mathcal{A} \to \mathbb{R}^+$ to sample $a \sim \pi_\theta(\cdot \mid z)$.
- 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 the policy 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 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 is online: it alternates training the model by minimizing the losses above, and collecting new data in the environment. At inference time, TD-MPC plans in the latent space with Model Predictive Control, 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) [38], 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 critic's value estimate at the last state. (3) One of the best n_e action sequences sampled in the last iteration of the previous step is randomly selected, and its first action is executed.

IV. IQL-TD-MPC for Offline Model-Based RL

In this section, we present IQL-TD-MPC, a framework that extends TD-MPC to the offline RL setting via Implicit Q-Learning (IQL) [14]. As shown in Section VI, naively training a 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. Indeed, 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.

In the original IQL work [14], the authors address the challenge of out-of-distribution actions using 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, $L_2^\tau(u) = \tau u^2$ for $u \geq 0$ and $(1-\tau)u^2$ for $u < 0$, with $\tau \in (0.5, 1)$ 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 [39], using a weighted behavioral cloning loss whose weights scale exponentially with the advantage:

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

with advantage $A_\theta(s,a) = Q_{\theta^-}(s,a) - V_\theta(s)$, sg the stop gradient operator, and $\beta > 0$ the inverse temperature.

To integrate IQL into TD-MPC (Section III-B), we first replace the TD-MPC policy 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. Indeed, MPC planning may still prefer actions that lead to high-return states under R_θ and Q_θ, while actually exploiting these models' blind spots and performing poorly. 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 III-B where the number of random action sequences n_r is set to zero.

But 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

[2]For convenience, when describing IQL, 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.
[3]Our policy implementation is based on the Soft Actor-Critic codebase [40].

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.

V. 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: the original TD-MPC paper [13] uses 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. To indicate this, we add the superscript M. The Manager processes trajectories $(s_0, a_0^M, r_k^M, s_k, a_k^M, \ldots, r_{kH}^M, s_{kH})$ 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 [33], [34]? Prior work learned an autoencoder that can reconstruct the next latent state [41], [20], and one could define the abstract action as its latent representation. 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. This model b_θ^M is trained implicitly by backpropagating through a_t the gradient of the total loss. Similar to Director [28], we found using discrete actions to stabilize training, and thus modify the policy π_θ^M to output discrete actions.

Once trained, the IQL-TD-MPC Manager can 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: $a_t^M = b_\theta^M(z_t^M, z_{t+k}^M)$. Eq. 5 is *not* indexed by k because intent embeddings are applied at each step.

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

Why are intent embeddings beneficial for offline RL? Before turning to experiments, we provide an intuitive explanation for why augmenting states with intent embeddings may be beneficial. For simplicity, we focus on the well-understood Behavioral Cloning (BC) algorithm, but we note that many other offline RL algorithms such as Advantage Weighted Actor-Critic (AWAC) [16], Implicit Q-Learning (IQL) [14], and Twin Delayed DDPG Behavioral Cloning (TD3-BC) [17] use the BC objective in some way, and thus the intuition may carry over to these algorithms as well.

We first provide an information-theoretic argument to explain why intent embedding should make the imitation learning loss easier to optimize. One key challenge in long-horizon sparse-reward offline RL is the ambiguity surrounding the relationship between a state-action pair in a dataset and its corresponding long-term objective. By incorporating intent embeddings derived from MPC planning at a coarser timescale 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 random variables (S_t, G_t) to a *Worker* action random variable A_t, with G_t the intent embedding derived from the *Manager* action. Since A_t is not independent of G_t given the state 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 intent embeddings may improve BC. This can be particularly beneficial on offline datasets built from a mixture of varied policies [15]. 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 (under R_θ^M and Q_θ^M when unrolling f_θ^M).

VI. 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 [15], following the experimental protocol from CORL [42]: 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 (both pre-training the Manager and training the Worker) took ~ 5 hours on average. We use $H = 4$ and $k = 8$ in the hierarchical settings, with little effort to tune these hyper-parameters as early experiments showed clear benefits of hierarchy with these values.

A. (Q1) IQL-TD-MPC performance in offline RL

We begin with a preliminary experiment to verify that IQL-TD-MPC is a viable offline RL algorithm, by comparing IQL-TD-MPC, TD-MPC, and several offline RL algorithms from the literature on various tasks. See Table I 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 maze2d umaze, where TD-MPC outperforms IQL-TD-MPC. We hypothesize that this is because the data provides adequate coverage for TD-MPC, while the conservative expectile updates of IQL-TD-MPC slow down learning. The other trend is that IQL-TD-MPC is generally on par with other offline RL algorithms, except on antmaze diverse datasets. One hypothesis for the poor performance on antmaze diverse is that the goal locations are random, leading to more diverse actions and thus higher-variance policies. This, in turn, may be "breaking" the planning stage due to overfitting on out-of-distribution actions.

B. (Q2) Benefits of using IQL-TD-MPC as a Manager

Now, we turn to the main results of our work, that demonstrate the benefits of using IQL-TD-MPC as a Manager with varied non-hierarchical offline RL algorithms as Workers (for this experiment, we used offline RL algorithms from the CORL repository [42]). We concatenated intent embeddings output by the Manager to the environment states seen by these Workers during both training and evaluation. The changes to the CORL algorithms were straightforward: (i) augment states in the offline dataset and (ii) wrap the evaluation environment to include intent embeddings.

Table II shows the results for the following CORL Workers: Advantage Weighted Actor-Critic (AWAC) [16], Behavioral Cloning (BC), Decision Transformer (DT) [18], Implicit Q-Learning (IQL) [14], Twin Delayed DDPG Behavioral Cloning (TD3-BC) [17], and Conservative Q-Learning (CQL) [19]. 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.

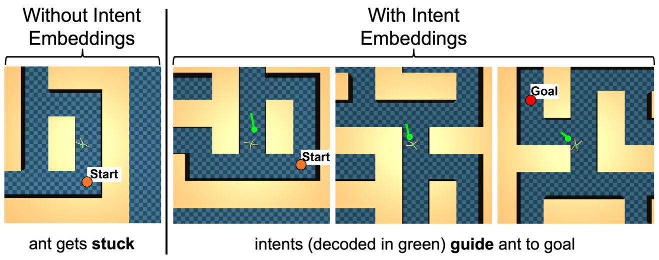
Without Intent Embeddings
With Intent Embeddings
Start
Start
Goal
ant gets **stuck**
intents (decoded in green) **guide** ant to goal

Fig. 2: Visualizing an episode of the Behavioral Cloning agent on antmaze-large-play-v2. 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. Video available at `https://sites.google.com/view/iql-td-mpc`.

Notably, our approach slightly worsens performance on 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. Intent embeddings are trained by having the Manager look at states k steps ahead, but this may not help on these tasks. We hypothesize that this may even hurt, by restricting the pool of candidate actions the Worker is considering.

In Fig. 2, we visualize an episode of the Behavioral Cloning (BC) agent on antmaze-large-play-v2, to qualitatively understand the benefits of our framework. On the left, without intent embeddings, the ant gets stuck close to the start of the maze. On the right, the ant reaches the goal, guided by the intent embeddings visualized in green. To generate these visualizations, we trained a decoder that converts intent embeddings (in the Manager's latent space) back into the raw environment state space, which contains the ant's position and velocity. The green dot shows the position, and the green line attached to it shows the velocity (speed proportional to length). This decoder was trained on a reconstruction loss and did not affect the training of the other models. The visualization shows that the intent embeddings act as latent-space subgoals exploited by the Worker.

C. (Q3) Impact of IQL-TD-MPC's intent embeddings

Could the strong results in Table II simply be due to the intent embeddings "tie-breaking" the stochasticity of the behavior policy? To address this question, in Table III we run our framework but replace the intent embeddings with random vectors of the same dimensionality, with entries drawn uniformly from $(0, 1)$. Across nearly all tasks and algorithms, we found no significant difference compared to the baseline, showing that the Workers learned to ignore the random vectors. Comparing against the clear benefits of our proposed method in Table II, we can conclude that IQL-TD-MPC was critical; it guides the Workers impactfully.

Interestingly, in Table III, Workers learned to ignore the random vectors in half-cheetah, while in Table II, our modification *harmed* performance. This confirms that the intent

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

TABLE I: **Preliminary experiment.** Normalized scores of IQL-TD-MPC, other offline RL algorithms (IQL [14], TT [23], TAP [24]) and TD-MPC on D4RL after 1M training steps. IQL results are from [42]. TT and TAP results are from their papers, except for antmaze-umaze-diverse and maze2d, which we reproduced with the default hyper-parameters since 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 best.

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 II: **Main experiment.** 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 [42]. We do not consider TAP because it includes goals in the observations, making the comparison unfair. Each table entry is of the form "baseline evaluation score → our evaluation score". We report scores after 500K steps of training; in general, agents plateaued after this point. For our hierarchical framework, these 500K steps correspond to 300K steps of pre-training the Manager, then 200K steps of training the CORL Worker. All entries report a mean over 5 independent random seeds; Green entries indicate statistically significant ($p < 0.05$) improvement, while 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	$95 \rightarrow 93$	$57 \rightarrow 61$	$92 \rightarrow 90$	$86 \rightarrow 87$

TABLE III: Ablation results, where we replace Manager intent embeddings with random vectors. Each table entry is of the form "baseline evaluation score → ablation evaluation score". We report scores after 500K steps of training. All entries report a mean over 5 independent random seeds; Red entries indicate statistically significant ($p < 0.05$) degradation.

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.

VII. Limitations and Future Work

In this paper, we proposed a non-trivial extension of TD-MPC to the offline setting based on IQL, and leveraged it as a temporally extended Manager in a hierarchical architecture.

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 II), 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 "distracted" by the 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. (3) Our Manager's timescale is defined by a fixed hyper-parameter k. This k could instead be set dynamically by the Manager, and included in the intent embeddings. (4) Similar to TD-MPC, our approach is computationally intensive as it involves unrolling the Manager's world model. Representing the world model as a Transformer [22], [5] may allow for more efficient rollouts. (5) Our approach implicitly assumes a deterministic environment, and would likely need further adjustments to work efficiently in a stochastic setting [43].

References

[1] D. Ha and J. Schmidhuber, "Recurrent world models facilitate policy evolution," in *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, ser. NeurIPS'18, 2018, pp. 2455–2467.

[2] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., 2019, pp. 2555–2565.

[3] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver, "Mastering Atari, Go, Chess and Shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, Dec 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-03051-4

[4] W. Ye, S. Liu, T. Kurutach, P. Abbeel, and Y. Gao, "Mastering atari games with limited data," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 25 476–25 488.

[5] V. Micheli, E. Alonso, and F. Fleuret, "Transformers are sample efficient world models," 2022. [Online]. Available: https://arxiv.org/abs/2209.00588

[6] A. Argenson and G. Dulac-Arnold, "Model-based offline planning," 2021.

[7] I. Clavera, V. Fu, and P. Abbeel, "Model-augmented actor-critic: Backpropagating through paths," 2020.

[8] M. Botvinick and A. Weinstein, "Model-based hierarchical reinforcement learning and human action control," *Philos Trans R Soc Lond B Biol Sci*, vol. 369, no. 1655, Nov. 2014.

[9] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[10] R. F. Prudencio, M. R. Maximo, and E. L. Colombini, "A survey on offline reinforcement learning: Taxonomy, review, and open problems," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[11] S. Lange, T. Gabel, and M. A. Riedmiller, "Batch reinforcement learning," in *Reinforcement Learning*, 2012.

[12] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *Journal of Machine Learning Research*, vol. 6, 2005.

[13] N. Hansen, X. Wang, and H. Su, "Temporal difference learning for model predictive control," *arXiv preprint arXiv:2203.04955*, 2022.

[14] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with Implicit Q-Learning," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=68n2s9ZJWF8

[15] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," 2020.

[16] A. Nair, A. Gupta, M. Dalal, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," 2020. [Online]. Available: https://arxiv.org/abs/2006.09359

[17] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 20 132–20 145.

[18] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021.

[19] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1179–1191.

[20] J. Li, C. Tang, M. Tomizuka, and W. Zhan, "Hierarchical planning through goal-conditioned offline reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 216–10 223, 2022.

[21] H. Xu, L. Jiang, J. Li, and X. Zhan, "A policy-guided imitation approach for offline reinforcement learning," 2022. [Online]. Available: https://arxiv.org/abs/2210.08323

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[23] M. Janner, Q. Li, and S. Levine, "Offline reinforcement learning as one big sequence modeling problem," in *Advances in Neural Information Processing Systems*, 2021.

[24] Z. Jiang, T. Zhang, M. Janner, Y. Li, T. Rocktäschel, E. Grefenstette, and Y. Tian, "Efficient planning in a compact latent action space," 2022. [Online]. Available: https://arxiv.org/abs/2208.10291

[25] A. van den Oord, O. Vinyals, and k. kavukcuoglu, "Neural discrete representation learning," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[26] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning latent plans from play," in *Proceedings of the Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds., vol. 100. PMLR, 30 Oct–01 Nov 2020, pp. 1113–1132.

[27] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015.

[28] D. Hafner, K.-H. Lee, I. Fischer, and P. Abbeel, "Deep hierarchical planning from pixels," 2022. [Online]. Available: https://arxiv.org/abs/2206.04114

[29] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[30] A. Bagaria and G. Konidaris, "Option discovery using deep skill chaining," in *International Conference on Learning Representations*, 2020.

[31] C. Daniel, H. Van Hoof, J. Peters, and G. Neumann, "Probabilistic inference for determining options in reinforcement learning," *Machine Learning*, vol. 104, pp. 337–357, 2016.

[32] E. Brunskill and L. Li, "Pac-inspired option discovery in lifelong reinforcement learning," in *International conference on machine learning*. PMLR, 2014, pp. 316–324.

[33] K. Pertsch, Y. Lee, and J. Lim, "Accelerating reinforcement learning with learned skill priors," in *Conference on robot learning*. PMLR, 2021, pp. 188–204.

[34] E. Rosete-Beas, O. Mees, G. Kalweit, J. Boedecker, and W. Burgard, "Latent plans for task-agnostic offline reinforcement learning," in *Conference on Robot Learning*. PMLR, 2023, pp. 1838–1849.

[35] Y. LeCun, "A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27," *Open Review*, vol. 62, 2022.

[36] M. L. Puterman, "Markov decision processes," *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.

[37] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[38] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control using covariance variable importance sampling," *CoRR*, vol. abs/1509.01149, 2015. [Online]. Available: http://arxiv.org/abs/1509.01149

[39] X. B. Peng, A. Kumar, G. Zhang, and S. Levine, "Advantage-weighted regression: Simple and scalable off-policy reinforcement learning," 2019. [Online]. Available: https://arxiv.org/abs/1910.00177

[40] D. Yarats and I. Kostrikov, "Soft actor-critic (sac) implementation in pytorch," 2020.

[41] A. Mandlekar, F. Ramos, B. Boots, S. Savarese, L. Fei-Fei, A. Garg, and D. Fox, "IRIS: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data," 2020. [Online]. Available: https://arxiv.org/abs/1911.05321

[42] D. Tarasov, A. Nikulin, D. Akimov, V. Kurenkov, and S. Kolesnikov, "CORL: Research-oriented deep offline reinforcement learning library," in *3rd Offline RL Workshop: Offline RL as a "Launchpad"*, 2022. [Online]. Available: https://openreview.net/forum?id=SyAS49bBcv

[43] I. Antonoglou, J. Schrittwieser, S. Ozair, T. K. Hubert, and D. Silver, "Planning in stochastic environments with a learned model," in *International Conference on Learning Representations*, 2022.