E-MCTS: Deep Exploration in Model-Based Reinforcement Learning by Planning with Epistemic Uncertainty

Yaniv Oren, Matthijs T. J. Spaan & Wendelin Böhmer
Delft University of Technology
2628 CD Delft, The Netherlands
{y.oren,m.t.j.spaan,j.w.bohmer}@tudelft.nl

Abstract

One of the most well-studied and highly performing planning approaches used in Model-Based Reinforcement Learning (MBRL) is Monte-Carlo Tree Search (MCTS). Key challenges of MCTS-based MBRL methods remain dedicated deep exploration and reliability in the face of the unknown, and both challenges can be alleviated through principled epistemic uncertainty estimation in the predictions of MCTS. We present two main contributions: First, we develop methodology to propagate epistemic uncertainty in MCTS, enabling agents to estimate the epistemic uncertainty in their predictions. Second, we utilize the propagated uncertainty for a novel deep exploration algorithm by explicitly planning to explore. We incorporate our approach into variations of MCTS-based MBRL approaches with learned and provided dynamics models, and empirically show deep exploration through successful epistemic uncertainty estimation achieved by our approach. We compare to a non-planning-based deep-exploration baseline, and demonstrate that planning with epistemic MCTS significantly outperforms non-planning based exploration in the investigated deep exploration benchmark.

1 Introduction

Model-based reinforcement learning (MBRL) has shown tremendous achievements in recent years, from super-human performance in games [Schrittwieser et al., 2020, Silver et al., 2018], to outperforming human designers in tasks that previously relied on intricate human engineering [Mandhane et al., 2022]. MBRL algorithms most commonly leverage their model (whether it is dynamically learned as part of the RL task, or pre-specified to the agent) for planning [Moerland et al., 2023]. Some of the best performing planning-based MBRL approaches, Mu/AlphaZero [Schrittwieser et al., 2020, Silver et al., 2018] rely on Monte-Carlo Tree Search (MCTS), a structured, extensively researched and commonly used planning approach. While the final performance that has been demonstrated with these algorithms is record-breaking, they are notoriously expensive to train, in compute as well as in samples. They are also unable to estimate the epistemic uncertainty in their predictions, preventing them from being reliable in the face of the unknown. A common approach for improving sample efficiency is through improved exploration. Exploration approaches range from uninformed random-action-selection based (such as employed by Alpha/MuZero) to more advanced approaches, such as exploration bonuses based on epistemic uncertainty[1], which incentivize future visitations to states and actions that are expected to result in new knowledge.

[1]Epistemic uncertainty is usually defined as being reducible with more observations. Planning with models and interaction with the environment yield therefore different epistemic uncertainties. In this paper we only refer to uncertainty that can be reduced by more exploration, i.e., the *epistemic uncertainty* of learned models.

16th European Workshop on Reinforcement Learning (EWRL 2023).

Effective epistemic uncertainty estimation in the predictions of the agent can be used to improve in both areas of challenge: reliability in the face of the unknown and advanced exploration. Further, planning for exploration promises to harness the benefits of planning (such as improved value estimation and better action selection) explicitly for exploration, an idea which was previously explored by Sekar et al. [2020] with promising results.

In this work, we develop methodology to 1) incorporate epistemic uncertainty into MCTS, enabling agents to estimate the epistemic uncertainty associated with predictions at the root of the MCTS planning tree (**E**pistemic-**MCTS**) and 2) leverage the uncertainty for deep exploration that capitalizes on the strengths of planning, by modifying the MCTS objective to an exploratory objective. We evaluate our agent on the benchmark hard-exploration task of Deep Sea [Osband et al., 2020] against exploration baselines that do not leverage planning. In our experiments, our agent demonstrates deep exploration and significantly outperforms both naive and sophisticated exploration baselines.

The remainder of this paper is organized as follows: Section 2 provides relevant background for MBRL, MCTS and epistemic uncertainty estimation in deep RL. Section 3 describes our contributions, starting with distinguishing between epistemic and non-epistemic sources in MCTS, followed by the framework for uncertainty propagation in MCTS (E-MCTS), our approach for harnessing E-MCTS to achieve deep exploration and finally a discussion regarding the challenges and limitations in estimating epistemic uncertainty in planning with an abstracted, learned model of the environment. Section 4 discusses related work. Section 5 evaluates our method with different dynamics models against a hard-exploration benchmark and compares to standard exploration baselines. Finally, Section 6 concludes the paper and discusses future work.

2 Background

In RL, an agent learns a behavior policy $\pi(a|s)$ through interactions with an environment, by observing states (or observations), executing actions and receiving rewards. The environment is represented with a Markov Decision Process [MDP, Bellman, 1957] , or a partially-observable MDP [POMDP, Åström, 1965]. An MDP $\mathcal{M}$ is a tuple: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \rho, P, R \rangle$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ a set of actions, ρ the initial state distribution, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ a bounded reward function, and $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition function, where $P(s_{t+1}|s_t, a_t)$ specifies the probability of transitioning from state s_t to state s_{t+1} after executing action a_t at time t. In a POMDP $\mathcal{M}' = \langle \mathcal{S}, \mathcal{A}, \rho, P, R, \Omega, O \rangle$, the agent observes observations $o_t \in \Omega$. $O : \mathcal{S} \times \mathcal{A} \times \Omega \to [0, 1]$ specifies the probability $O(o|s_t, a_t)$ of observing o. In MBRL the agent uses a model of the environment to optimize its policy, often through planning. The model is either learned from interactions, or provided. In Deep MBRL (DMBRL) the agent utilizes deep neural networks as function approximators. Many RL approaches rely on learning a state-action *Q-value function* $Q^\pi(s, a) = \mathbb{E}[R(s, a, s') + \gamma V^\pi(s')|\, {}_{s' \sim P(\cdot|s,a)}]$ or the corresponding state *value function* $V^\pi(s) = \mathbb{E}[Q^\pi(s, a)|\, {}_{a \sim \pi(\cdot|s)}]$, which represents the expected return from starting in state s (and possibly action a) and then following a policy $\pi(a_t|s_t)$ which specifies the probability of selecting the action a_t in state s_t. The discount factor $0 < \gamma < 1$ is used in infinite-horizon (PO)MDPs to guarantee that the values remain bounded, and is commonly used in RL for learning stability.

2.1 Monte Carlo Tree Search

MCTS is a planning algorithm that constructs a planning tree with the current state s_t at its root to estimate the objective: $\arg\max_a \max_\pi Q^\pi(s_t, a)$. The algorithm iteratively performs *trajectory selection*, *expansion*, *simulation* and *backup* to arrive at better estimates at the root of the tree. At each planning step i, starting from the root node $s^i_{t,0} \equiv \hat{s}_0$, the algorithm selects a trajectory in the existing tree based on the averaged returns $q(\hat{s}_k, a)$ experienced in past trajectories selecting the action a in the same node $\hat{s}_k$, and a search heuristic, such as an Upper Confidence Bound for Trees [UCT, Kocsis and Szepesvári, 2006]:

$$a_k \;=\; \arg\max_{a \in A} q(\hat{s}_k, a) + 2C_p \sqrt{\frac{2 \log(\sum_{a'} N(\hat{s}_k, a'))}{N(\hat{s}_k, a)}}, \tag{1}$$

where $N(\hat{s}_k, a)$ denotes the number of times action a has been executed in node $\hat{s}_k$, and $C_p > 0$ trades off exploration of new nodes with maximizing observed return. When the the trajectory selection arrives at a leaf node $\hat{s}_T$ MCTS expands the node and estimates its initial value as the average of Monte-Carlo rollouts using a random policy. Recent DMBRL algorithms that use MCTS such as Alpha/MuZero [Silver et al., 2016, 2017, 2018, Schrittwieser et al., 2020] replace the rollouts with a value function $v(\hat{s}_T)$ that is approximated by a neural network and use the PUCT [Rosin, 2011] search heuristic instead of UCT:

$$a_k \;=\; \arg\max_{a \in A} q(\hat{s}_k, a) + \pi(a|\hat{s}_k)\, C_p \frac{\sqrt{\sum_{a'} N(\hat{s}_k, a')}}{1 + N(\hat{s}_k, a)}. \tag{2}$$

Where $\pi(a|\hat{s}_k)$ is either given, or learned by imitating the MCTS policy π^{MCTS}, to incorporate prior knowledge into the search. MCTS propagates the return (discounted reward for visited nodes plus leaf's value) back along the planning trajectory. At the root of the tree, the optimal value $\max_\pi V^\pi(s_t)$ of current state s_t is estimated based on the averaged returns experienced through every action a, and averaged over the actions:

$$\max_\pi V^\pi(s_t) \;\approx\; \sum_{a \in \mathcal{A}} \frac{N(\hat{s}_0, a)}{\sum_{a' \in \mathcal{A}} N(\hat{s}_0, a')}\, q(\hat{s}_0, a) \;\; =: \sum_{a \in \mathcal{A}} \pi^{\text{MCTS}}(a|s_t)\, q(\hat{s}_0, a) \;\; =: \; v_t^{\text{MCTS}}. \tag{3}$$

2.2 MCTS-Based MBRL

MCTS requires access to three core functions. Those are: (i) a representation function $g(s_t) = \hat{s}_0 \in \hat{\mathcal{S}}$ that encodes the current state at the root of the tree into a latent space, in which (ii) a transition function $f(\hat{s}_k, a_k) = \hat{s}_{k+1}$ predicts the next latent state and (iii) a function $r(\hat{s}_k, a_k) = \mathbb{E}[r_k|\hat{s}_k, a_k]$ that predicts the corresponding average reward. Such models in an latent state space $\hat{\mathcal{S}} \neq \mathcal{S}$ do not have to distinguish between different true states $s, s' \in \mathcal{S}$, i.e., $g(s) = g(s'), s \neq s'$, if such a distinction does not benefit value and reward prediction, and are commonly called *value-equivalent* or *abstracted* models. Note that for an identity function $g(s_t) = s_t$ all models, functions and policies would be defined in the true state space $\mathcal{S}$, and that in a POMDP g can encode the current observation o_t or the entire action-observation history $\langle o_0, a_0, o_1, a_1, \ldots, o_t \rangle$. As in Mu/AlphaZero [Schrittwieser et al., 2020, Silver et al., 2018], a value function $v(\hat{s}_T)$ can be learned for replacing rollouts, and a policy function $\pi(a|\hat{s}_k)$ imitates the MCTS policy to bias planning towards promising actions based on prior knowledge. In deep MBRL (DMBRL) these functions are learned with deep neural networks.

Five common learning signals are used to train the transition model f with varying horizons k:
1) A reconstruction loss $L^k_{re}\big(h(\hat{s}_k), s_{t+k}\big)$, training a decoder h to reconstruct true states s_{t+k} from latent representations $\hat{s}_k$ that have been predicted from $\hat{s}_0 = g(s_t)$, shaping both g and f.
2) A consistency loss $L^k_{\text{co}}\big(\hat{s}_k, g(s_{t+k})\big)$, training the model that predicted states should align with latent representation of states s_t (or observations/histories in POMDP). Critically, L^k_{co} is not used to train g, only f. When the representation function g is an identity, L^k_{re} and L^k_{co} can be thought of as providing the same learning signal. Otherwise, they can be used independently or in combination.
3) A reward loss $L^k_r\big(r(\hat{s}_k, a_k), r_{t+k}\big)$, where the model is trained to predict representations that enable predictions of, and are aligned with, the true rewards observed in the environment r_t.
4) A value loss $L^k_v\big(v(\hat{s}_k), v^{\text{MCTS}}_{t+k}\big)$ that similarly trains the model to predict states that enable value learning.
5) A policy loss $L^k_\pi\big(\pi(\cdot|\hat{s}_k), \pi^{\text{MCTS}}(\cdot|s_{t+k})\big)$ that trains prior policy π to predict the MCTS policy. These losses are described in more detail in Appendix B.2.

2.3 Estimating Epistemic Uncertainty in Deep Reinforcement Learning

Predictive epistemic uncertainty refers to any uncertainty that is associated with a prediction and is sourced in lack-of-information. For example, prior to repeated tosses of a coin, there can be high uncertainty whether the coin is fair or not. The more the coin has been tossed, the more certain we can be about the coin's fairness, even if we will always retain uncertainty in the exact prediction of heads or tails, without access to a precise simulation of the physics of the coin toss (referred to as *aleatoric* uncertainty, or the inherent uncertainty in the way we choose to model a coin). Defining, quantifying and estimating predictive epistemic uncertainty is an active field of research that encompasses many approaches and many methods (see [Hüllermeier and Waegeman, 2021, Lockwood and Si, 2022]). In this work, we take the common approach for quantifying epistemic uncertainty as the variance in a probability distribution of predictions that are consistent with observations $\text{Var}_X(X|s_t) = \mathbb{V}[X|s_t]$.

As for estimating epistemic uncertainty, two standard approaches are the distributional approach and the proxy-based approach. The distributional approach approximates a probability distribution over possible predictions with respect to the agent's experiences, while the proxy-based approach aims to directly predict a measure for *novelty* of experiences. Two reliable and lightweight methods for novelty-based epistemic uncertainty estimation are Random Network Distillation (RND) [Burda et al., 2019] and state-visitation counting. RND evaluates novelty as the difference between the prediction of a randomly initialized untrained target network ψ' and a to-be trained network ψ with a similar architecture. The network ψ is trained to match the predictions of the target network for the observed states (or state-action pairs) with MSE loss $L_{rnd}\big(\psi(s_t, a_t), \psi'(s_t, a_t)\big) = ||\psi(s_t, a_t) - \psi'(s_t, a_t)||^2$. Novel observations are expected to produce unpredictable outputs from the target network, and thus the difference between the prediction of the target network and the trained network serves as a proxy-measure for novelty. These methods encapsulate the epistemic uncertainty in a local prediction: for example, uncertainty in prediction of reward or next state. Estimating epistemic uncertainty in value predictions that contain the uncertainty that propagates from future decisions made by a policy is a different matter. One method to estimate value uncertainty is the Uncertainty Bellman Equation [UBE, O'Donoghue et al., 2018]. UBE approximates an upper bound on the epistemic uncertainty in value (here interpreted as variance of the Q-value) as the sum of local uncertainties $\sigma^2(s_t, a_t)$ that are associated with the decisions a_t at states s_t:

$$U^{\pi}(s_t) \;:=\; \mathbb{E}_{\pi}\Bigg[\sum_{i=0}^{\infty}\gamma^{2i}\sigma^2(s_{t+i}, a^{\pi}_{t+i})\Bigg] \;=\; \mathbb{E}_{\pi}\Bigg[\sum_{i=0}^{n-1}\gamma^{2i}\sigma^2(s_{t+i}, a^{\pi}_{t+i}) + \gamma^{2n}U^{\pi}(s_{t+n})\Bigg].$$

In other words, UBE proposes to approximate the value uncertainty as the sum of twice-discounted local uncertainties and learn it with (possibly n-step) TD targets in a similar manner to value learning.

3 Deep Exploration by Epistemic MCTS

We begin by identifying different sources of uncertainty in MCTS with function approximation (section 3.1). We follow with our method to propagate epistemic uncertainty while circumventing the use of stochastic models (E-MCTS, section 3.2). We proceed to harness the propagated uncertainty for an optimistic planning objective to achieve deep exploration (section 3.3). Finally, we discuss challenges with estimating epistemic uncertainty when planning with latent models and possible solutions.

3.1 Sources of Uncertainty in MCTS

In this work we are concerned with estimating and leveraging the magnitude of *epistemic* uncertainty, to drive exploration in the environment. We begin by distinguishing between epistemic and non-epistemic sources of uncertainty in MCTS. In standard MCTS, the uncertainty in value prediction at each node stems from stochasticity in the environment and in the rollout policy (aleatoric). There are no learned quantities, and as such, there is no epistemic uncertainty. When a function approximator $v(s_t)$ is used to replace rollouts [such as in AlphaZero, Silver et al., 2018] the aleatoric uncertainty from MC rollout is replaced by uncertainty in the value prediction $v(s_t)$. We distinguish between two sources of uncertainty about $v(s_t)$: 1. Epistemic sources: errors resulting from evaluating $v(s_t)$ on unobserved states s_t. 2. Non-epistemic sources: approximation errors, TD-errors, stochasticity of the environment, stochasticity of the policy, model-class errors, and every other source of uncertainty that will not reduce directly by training on additional unique interactions with the environment. MCTS addresses non-epistemic uncertainty by averaging over node values, but does not address epistemic uncertainty. When function approximators are used to learn a model of the environment transition $f(s_t, a_t)$ and/or reward dynamics $r(s_t, a_t)$ [such as in MuZero, Schrittwieser et al., 2020] the uncertainty in value prediction will contain the uncertainty in the learned dynamics f, r. The same separation between epistemic and non-epistemic sources of uncertainty in f, r can be made. Distinguishing between epistemic and non-epistemic uncertainty allows us to concentrate on propagating only *epistemic* uncertainty, which can help the agent to find unobserved states that are worth exploring. In the following section we 1) investigate how to estimate and 2) develop a method to propagate the epistemic component of the uncertainty, and in the later sections we show how the epistemic uncertainty can be leveraged in MCTS for deep exploration following the principle of optimism in the face of uncertainty.

3.2 Propagating Uncertainty in MCTS

At planning step i, selecting a path of length T through a decision tree is equivalent to choosing a sequence of T actions $a^i_{0:T-1}$ that start at node $\hat{s}^i_0 = g(s_t)$ and end up in a leaf node $\hat{s}^i_T$. Deterministic models f, r predict the transitioned to nodes $\hat{s}_k^i$ and the encountered rewards r_k^i in nodes $\hat{s}_k^i, 0 \leq k < T$, respectively. The value v_T^i at leaf $\hat{s}_T^i$ is predicted by Monte-Carlo rollouts with f or directly with a neural network v. The values and rewards are used to update the n-step discounted return ν_k^i of each node $\hat{s}_k^i$ on the selected path:

$$\nu_k^i \quad := \quad \sum_{j=k}^{T-1} \gamma^{j-k} r_j^i + \gamma^{T-k} v_T^i \quad = \quad r_k^i + \gamma \nu_{k+1}^i \,, \qquad 0 \leq k < T \,, \qquad \nu_T^i = v_T^i, \tag{4}$$

where γ^{j-k} is the discount factor to the power of $j - k$ and the superscript i is indexing the planning step. Our following analysis is done per planning step i and we will drop the index i for the sake of readability. If (any of) f, r, v are assumed to be inexact r_k and v_T can be modelled as random variables in a Markov chain that is connected by random state-variables. The stochasticity in the chain captures the uncertainty in f, r, v's predictions. To clarify notation, we will refer to these as random states $\hat{S}_k$, rewards R_k, values V_k and returns $\mathcal{V}_k$. In line with the optimistic exploration literature, we aim to incentivize choosing actions in the environment associated with paths in the planning tree that have *epistemically* uncertain returns $\mathcal{V}_0$ in order to seek new high-reward interactions. For this we need to estimate the *epistemic* variance (variance from epistemic sources) $\mathbb{V}[\mathcal{V}_0 | s_t, a_{0:T-1}] \equiv \mathbb{V}[\mathcal{V}_0]$ of the return along a selected path $a_{0:T-1}$, starting with state s_t. To circumvent having to replace f, r, v with an explicitly stochastic model to propagate the uncertainty, we instead develop a direct and computationally efficient approximation for $\mathbb{V}[\mathcal{V}_0]$.

We will begin by deriving the mean and variance of the distribution of state-variables in the Markov chain for a given sequence of actions $a_{0:T-1}$. Let us assume we are given a differentiable transition function $f(\hat{S}_k, a_k) := \mathbb{E}_{\hat{S}_{k+1}}[\hat{S}_{k+1} | \hat{S}_k, a_k] \in \mathbb{R}^{|\hat{\mathcal{S}}|}$, which predicts the conditional expectation over the next state, and a differentiable uncertainty function $\mathbf{\Sigma}(\hat{S}_k, a_k) := \mathbb{V}_{\hat{S}_{k+1}}[\hat{S}_{k+1} | \hat{S}_k, a_k] \in \mathbb{R}^{|\hat{\mathcal{S}}| \times |\hat{\mathcal{S}}|}$ that yields the conditional-covariance matrix of the distribution. In DMBRL the assumption that models are differentiable is standard (see Section 2.2). We assume that the mean $\hat{s}_0$ of the first state-variable $\hat{S}_0$ is given as an encoding function $\hat{s}_0 = \mathbb{E}[\hat{S}_0 | s_t] = g(s_t)$, like in MuZero. The mean $\hat{s}_{k+1}$ of a later state-variable $\hat{S}_{k+1}$ can be approximated with a first order Taylor expansion around the previous mean $\hat{s}_k := \mathbb{E}[\hat{S}_k]$:

$$\begin{aligned} \hat{s}_{k+1} \quad &:= \quad \mathbb{E}[\hat{S}_{k+1}] \quad = \quad \mathbb{E}_{\hat{S}_k}[\mathbb{E}_{\hat{S}_{k+1}}[\hat{S}_{k+1} | \hat{S}_k, a_k]] \quad = \quad \mathbb{E}[f(\hat{S}_k, a_k)] \\ &\approx \quad \mathbb{E}[f(\hat{s}_k, a_k) + (\hat{S}_k - \hat{s}_k)^\top \nabla_{\hat{s}} f(\hat{S}, a_k)|_{\hat{S}=\hat{s}_k}] \quad = \quad f(\hat{s}_k, a_k) \,. \end{aligned} \tag{5}$$

In other words, under the assumption that the model f predicts the *expected* next state we reinterpret the original latent state $\hat{s}_k$ as the mean of the uncertain state $\mathbb{E}[\hat{S}_k]$.

To approximate the covariance $\mathbf{\Sigma}_{k+1} := \mathbb{V}[\hat{S}_{k+1}]$ or *the total uncertainty associated with state variable* $\hat{S}_{k+1}$ we need the *law of total variance*. The law of total variance states that for two random variables X and Y holds $\mathbb{V}[Y] = \mathbb{E}_X \left[\mathbb{V}_Y[Y|X] \right] + \mathbb{V}_X \left[\mathbb{E}_Y[Y|X] \right]$ (see Appendix A for a proof in our notation). Using the law of total variance and again a first order Taylor approximation around the previous mean state $\hat{s}_k$:

$$\begin{aligned} \mathbf{\Sigma}_{k+1} \quad := \quad \mathbb{V}[\hat{S}_{k+1}] \quad &= \quad \underbrace{\mathbb{E}_{\hat{S}_k}\left[\mathbb{V}_{\hat{S}_{k+1}}[\hat{S}_{k+1} | \hat{S}_k, a_k]\right]} \quad + \quad \underbrace{\mathbb{V}_{\hat{S}_k}\left[\mathbb{E}_{\hat{S}_{k+1}}[\hat{S}_{k+1} | \hat{S}_k, a_k]\right]} \\ &\approx \quad \qquad \mathbf{\Sigma}(\hat{s}_k, a_k) \qquad \quad + \quad \boldsymbol{J}_f(\hat{s}_k, a_k) \, \mathbf{\Sigma}_k \, \boldsymbol{J}_f(\hat{s}_k, a_k)^\top \,. \end{aligned} \tag{6}$$

Note that $f(\hat{S}_k, a_k) - \mathbb{E}[f(\hat{S}_k, a_k)] \approx (\hat{S}_k - \hat{s}_k)^\top \nabla_{\hat{S}} f(\hat{S}, a_k)|_{\hat{S}=\hat{s}_k} =: (\hat{S}_k - \hat{s}_k)^\top \boldsymbol{J}_f(\hat{s}_k, a_k)^\top$, where $\boldsymbol{J}_f(\hat{s}_k, a_k)$ denotes the Jacobian matrix of function f at mean state $\hat{s}_k$ and action a_k. Using these state statistics, we can derive the means and variances of causally connected variables like rewards R_k and values V_T. We assume that the conditional reward distribution has conditional mean $r(\hat{S}_k, a_k) := \mathbb{E}_{R_k}[R_k | \hat{S}_k, a_k]$ and conditional variance $\sigma_R^2(\hat{S}_k, a_k) := \mathbb{V}_{R_k}[R_k | \hat{S}_k, a_k]$, and that the conditional value distribution has conditional mean $v(\hat{S}_T) := \mathbb{E}_{V_T}[V_T | \hat{S}_T]$ and conditional variance $\sigma_V^2(\hat{S}_T) := \mathbb{V}_{V_T}[V_T | \hat{S}_T]$. Analogous to above we can derive:

$$r_k := \mathbb{E}[R_k] \approx r(\hat{s}_k, a_k) \,, \qquad \mathbb{V}[R_k] \approx \sigma_R^2(\hat{s}_k, a_k) + \boldsymbol{J}_r(\hat{s}_k, a_k) \, \mathbf{\Sigma}_k \, \boldsymbol{J}_r(\hat{s}_k, a_k)^\top \,, \tag{7}$$

$$v_T := \mathbb{E}[V_T] \approx v(\hat{s}_T) \,, \qquad \mathbb{V}[V_T] \approx \sigma_V^2(\hat{s}_T) + \boldsymbol{J}_v(\hat{s}_T) \, \mathbf{\Sigma}_T \, \boldsymbol{J}_v(\hat{s}_T)^\top \,. \tag{8}$$

If we assume that R_k and the n-step return $\mathcal{V}_{k+1}$ from Equation 4 are independent, we can compute

$$\mathbb{E}[\mathcal{V}_k] \quad = \quad \mathbb{E}_{R_k,\mathcal{V}_{k+1}}[R_k + \gamma \mathcal{V}_{k+1}] \;=\; \mathbb{E}[R_k] + \gamma\,\mathbb{E}[\mathcal{V}_{k+1}]\,, \quad \mathbb{E}[\mathcal{V}_T] = \mathbb{E}[V_T] \tag{9}$$

$$\mathbb{V}[\mathcal{V}_k] \quad = \quad \mathbb{V}_{R_k,\mathcal{V}_{k+1}}[R_k + \gamma \mathcal{V}_{k+1}] \;=\; \mathbb{V}[R_k] + \gamma^2\,\mathbb{V}[\mathcal{V}_{k+1}]\,, \quad \mathbb{V}[\mathcal{V}_T] = \mathbb{V}[V_T] \tag{10}$$

We can therefore approximate the variance $\mathbb{V}[\mathcal{V}_0|s_t, a_{0:T-1}]$ using one (E-)MCTS search, expansion and back-propagation steps through the selected path $a_{0:T-1}$, similar to the value-estimation $\mathbb{E}\left[\mathcal{V}_0|s_t, a_{0:T-1}\right]$ that is being done by standard MCTS (see pseudo-code in Algorithm 1). When applying this approach to model-learning algorithms such as MuZero, we interpret the representation g, dynamics f, value v and reward r functions as outputting the conditional means $\hat{s}_0, \hat{s}_k, v_T, r_k$ respectively. When applying this approach to methods that learn only some of f, r, v (for example AlphaZero Silver et al. [2018] which learns only v) the predictions from unlearned components will be associated with epistemic uncertainty $= 0$. E-MCTS will propagate the epistemic uncertainty in the learned components according to the remaining nonzero terms in Equations 6, 7, 8, 10. Finally we note that while E-MCTS is designed with epistemic uncertainty of the learned models in mind, any source of uncertainty can be propagated with E-MCTS, so long as it is interpreted as the local variances in state, reward and value predictions (Equations 6, 7 and 8 respectively).

3.3 Planning for Exploration with MCTS

The UCT operator of MCTS takes into account uncertainty about a node's subtree via the visitation count (see Equation 1) to drive exploration *inside* the planning tree. To drive exploration in the environment, we add the environmental epistemic uncertainty into the UCT formula in a similar manner, as the averaged standard deviation:

$$a_k \quad := \quad \arg\max_a q(\hat{s}_k, a) + \beta\sqrt{\sigma_q^2(\hat{s}_k, a_k)} + 2C\sqrt{\frac{2\log(\sum_{a'} N(\hat{s}_k, a'))}{N(\hat{s}_k, a_k)}}\,, \tag{11}$$

where $\beta \geq 0$ is a constant that can be tuned per task to encourage more or less exploration. The term

$$\sigma_q^2(\hat{s}_k, a_k) \quad := \quad \mathbb{V}[R_k] + \frac{1}{N(\hat{s}_k, a_k)} \sum_{i=1}^{N(\hat{s}_k, a_k)} \mathbb{V}[\mathcal{V}_{k+1}^i] \tag{12}$$

sums the variances computed individually at every backup step i through the node that is reached by executing action a_k in latent state $\hat{s}_k$ using equations 7 and 10. At each backup step i, with actions a_k^i, state means $\hat{s}_k^i$ and covariances $\boldsymbol{\Sigma}_k^i$, the variance $\mathbb{V}[\mathcal{V}_k^i]$ is approximated based on equations 10 and 7:

$$\mathbb{V}[\mathcal{V}_k^i] \approx \sigma_R^2(\hat{s}_k^i, a_k^i) + \boldsymbol{J}_r(\hat{s}_k^i, a_k^i)\boldsymbol{\Sigma}_k^i \boldsymbol{J}_r(\hat{s}_k^i, a_k^i)^\top + \gamma^2 \mathbb{V}[\mathcal{V}_{k+1}^i]\,. \tag{13}$$

At every backup step we compute the variance at the leaf node (equation 8), which is then used to update the parent's variance along the trajectory iteratively using equation 13. Pseudo-code can be found in Algorithm 1, where the modifications introduced to MCTS are marked in blue.

When using other search heuristics such as PUCT or the extension of PUCT used in Gumbel MuZero [Danihelka et al., 2022, Grill et al., 2020] we propose to view the term $q(\hat{s}_k, a) + \beta\sqrt{\sigma_q^2(\hat{s}_k, a_k)}$ as an exploratory-Q-value-estimate (or epistemically-optimistic-Q-value estimate) and use it in place of $q(\hat{s}_k, a)$ to modify the planning objective into the exploratory objective. Once the MCTS-based search with respect to the exploratory Q-value has completed, action selection in the environment can be done in the same manner as for exploitation. For example, by sampling actions with respect to the visitation counts of each action at the root of the tree as done by the original MuZero.

3.4 Limitations in Estimating Epistemic Uncertainty in Planning

Epistemic uncertainty estimation techniques in RL are designed to evaluate uncertainty on predictions in the true observations from the environment [Osband et al., 2018, Burda et al., 2019]. These methods translate naturally into planning with transition models that operate over the *true* state space of the environment $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ (such as AlphaZero). When the latent state space $\hat{\mathcal{S}}$ is not identical to the true state space $\mathcal{S}$, however, novelty estimated in latent space may not reflect the novelty in the true state space. Specifically, value-equivalent models (such as used by MuZero, or any otherwise abstracted models) in sparse-reward environments may suffer from representation collapse by abstracting all states into one constant representation. As a result, all latent states may be associated with the same novelty of zero. To the best of our knowledge, no uncertainty estimation method exists that can reliably estimate the novelty of s based on such abstracted representations

Algorithm 1 E-MCTS, requires functions g, f, r, v and uncertainty estimators $\mathbf{\Sigma}, \sigma_R^2, \sigma_V^2$

function EMCTS(state s_t, β) $\triangleright$ $\beta = 0$ for unmodified MCTS exploitation episodes
  while within computation budget **do**
    SELECT($g(s_t)$, β) $\triangleright$ traverses tree from root $\hat{s}_0 = g(s_t)$ and adds new leaf
  return action a drawn from $\pi(a|s_t) = \frac{N(\hat{s}_0,a)}{\sum_{a'} N(\hat{s}_0,a')}$ $\triangleright$ MCTS action selection

function SELECT(node $\hat{s}_k$, β)
  $a_k \leftarrow \arg\max_a q(\hat{s}_k, a) + \beta\sqrt{\sigma_q^2(\hat{s}_k, a)} + \sqrt{\frac{2\log(\sum_{a'} N(\hat{s}_k,a'))}{N(\hat{s}_k,a)}}$ $\triangleright$ Equation 11
  if a_k already expanded **then** SELECT($f(\hat{s}_k, a_k)$, β) $\triangleright$ traverses tree
  else EXPAND($\hat{s}_k, a_k$) $\triangleright$ adds new leaf

function EXPAND(node $\hat{s}_k$, not yet expanded action a_k)
  $\hat{s}_{k+1}, \mathbb{E}[V_{k+1}] \leftarrow$ Execute unmodified MCTS expansion that creates a new leaf $\hat{s}_{k+1}$
  $\mathbf{\Sigma}_{k+1} \leftarrow \mathbf{\Sigma}(\hat{s}_k, a_k) + \boldsymbol{J}_f(\hat{s}_k, a_k)\, \mathbf{\Sigma}_k\, \boldsymbol{J}_f(\hat{s}_k, a_k)^\top$ $\triangleright$ node attribute of $\hat{s}_{k+1}$, Equation 6
  $\mathbb{V}[R_k] \leftarrow \sigma_R^2(\hat{s}_k, a_k) + \boldsymbol{J}_r(\hat{s}_k, a_k)\, \mathbf{\Sigma}_k\, \boldsymbol{J}_r(\hat{s}_k, a_k)^\top$ $\triangleright$ node attribute of $\hat{s}_{k+1}$, Equation 7
  $\mathbb{V}[V_{k+1}] \leftarrow \sigma_V^2(\hat{s}_{k+1}) + \boldsymbol{J}_v(\hat{s}_{k+1})\, \mathbf{\Sigma}_{k+1}\, \boldsymbol{J}_v(\hat{s}_{k+1})^\top$ $\triangleright$ Equation 8
  BACKUP($\hat{s}_{k+1}, \mathbb{E}[V_{k+1}], \mathbb{V}[V_{k+1}]$) $\triangleright$ updates the tree values & variances

function BACKUP(node $\hat{s}_{k+1}$, return-mean $\mathbb{E}[\mathcal{V}_{k+1}]$, return-uncertainty $\mathbb{V}[\mathcal{V}_{k+1}]$)
  $\hat{s}_k, a_k, \mathbb{E}[\mathcal{V}_k] \leftarrow$ Execute unmodified MCTS backup step (updates $q(\hat{s}_k, a_k)$ and $N(\hat{s}_k, a_k)$)
  $\mathbb{V}[\mathcal{V}_k] \leftarrow \mathbb{V}[R_k] + \gamma^2 \mathbb{V}[\mathcal{V}_{k+1}]$ $\triangleright$ uses node-attribute $\mathbb{V}[R_k]$, Equation 10
  $\sigma_q^2(\hat{s}_k, a_k) \leftarrow \sigma_q^2(\hat{s}_k, a_k) + \frac{\mathbb{V}[\mathcal{V}_k] - \sigma_q^2(\hat{s}_k, a_k)}{N(\hat{s}_k, a_k)}$ $\triangleright$ node attribute of $\hat{s}_{k+1}$, Equation 12
  if $k > 0$ **then** BACKUP($\hat{s}_k, \mathbb{E}[\mathcal{V}_k], \mathbb{V}[\mathcal{V}_k]$) $\triangleright$ updates the tree values & variances

$\hat{s}$. This problem can be circumvented however by driving reconstruction losses through the model, incentivizing the learned model to distinguish between unique states. Alternatively, an auxiliary dynamics model can be learned which does not need to be accurate or robust but only distinguish between novel states and observed states based on starting states and subsequently executed action sequences.

To benefit from the fact that E-MCTS does not require the epistemic uncertainty to be captured in a probabilistic model, when $\hat{\mathcal{S}} \approx \mathcal{S}$ we use the lightweight novelty estimator RND (see Section 2.3). RND is used over latent state action pair $(\hat{s}_k, a_k)$ as a proxy for the local variance $\sigma_R^2(\hat{s}_k, a_k)$. As RND does not explicitly model covariance matrices, we estimate the uncertainty in latent state $\mathbf{\Sigma}_k$ and the uncertainty in the predictions based on latent state σ_R together (see Appendix B.3). We show that this choice is sufficient for E-MCTS to significantly improve over a comparable non-planning deep exploration baseline in Section 5. For the value-equivalent dynamics model of MuZero we provide the agent with reliable (but unrealistic) transition uncertainty in the form of state-action visitation counts in the true state space $\mathcal{S}$, and use it in a similar manner, to evaluate E-MCTS in the presence of reliably uncertainty and abstracted models.

To estimate the value uncertainty at the leaf $\sigma_V^2(\hat{s}_T)$ we use UBE (see Section 2.3) for all three model cases. UBE is natural to use in the anchored and true model cases, by simply adding a UBE prediction head u to the Mu/AlphaZero architecture, which approximates $u(\hat{s}_k) \approx U^\pi(s_k)$. A possible alternative to UBE is to train an ensemble of value functions and verify that the value targets for each ensemble member i is a target network based on the same ensemble member i (i.e., not use the mean of the ensemble, or the value of the root of an MCTS tree as target). This choice is important to retain diversity in the ensemble predictions for uncertain states even after value-bootstrapping in TD learning. In the abstracted model case training the model to enable UBE predictions (similar to the way in which the model is trained to enable value predictions, see Section 2.2), enables the agent to estimate value-uncertainty in planning (see Section 5).

4 Related Work

Different faces of the idea of leveraging planning with learned dynamics models for exploration have been investigated by a range of previous works, such as Yi et al. [2011], Hester and Stone [2012], Shyam et al. [2019], Sekar et al. [2020], Lambert et al. [2022] and Henaff [2019]. Among a range of differences, these methods are not tailored for MCTS or deterministic dynamics's models MCTS algorithms, which are a very strong class of MBRL algorithms. We add to this line of work E-MCTS: tailored for MCTS (and planning trees in general), lightweight and applicable to deterministic models by approximating and propagating the variance directly resulting only in a constant increase in computation cost to MCTS. Moerland et al. [2020] identify that the further a state is from a terminal state in the MCTS planning tree, the more uncertainty should be associated with it in planning, and utilizes this uncertainty to bias search in MCTS. POMCP [Silver and Veness, 2010], POMCPOW [Sunberg and Kochenderfer, 2018] and BOMCP [Mern et al., 2021] extend MCTS to POMDPs with a probabilisticly modelled Bayesian belief state at the nodes using a probabilistic model, while Stochastic MuZero Antonoglou et al. [2021] extended MuZero to the stochastic setting by replacing f with a Vector Quantised Variational AutoEncoder [van den Oord et al., 2017]. Epistemic uncertainty is not distinguished explicitly or used for exploration. A common uncertainty / novelty estimation alternative to RND Burda et al. [2019] are ensembles Lakshminarayanan et al. [2016], Ramesh et al. [2022]. The uncertainty measure is usually the disagreement between the ensemble's predictions. Bootstrapped DQN [BDQN, Osband et al., 2016, 2018] is an effective model-free deep exploration approach that relies on the epistemic uncertainty estimated by an ensemble to drive exploration. Wasserstein Temporal Difference [WTD, Metelli et al., 2019] offers an alternative to UBE O'Donoghue et al. [2018] for propagating epistemic uncertainty in TD-learning, using Wasserstein Barycenters Agueh and Carlier [2011] to update a posterior over Q functions in place of a standard Bayesian update. In addition, UBE was criticized by Janz et al. [2019] for having unnecessary properties as well as being insufficient for deep exploration with posterior-sampling based RL [PSRL, Osband et al., 2013]. These shortcomings however do not influence UCB-based exploration algorithms which E-MCTS can be classified as. Pairing with UBE thus enables E-MCTS to benef from the beneficial properties of UBE (such as uncertainty propagation, as discussed by Janz et al. [2019]) while avoiding the shortcomings identified by Janz et al. [2019].

5 Experiments

We evaluate the following hypotheses: 1. E-MCTS successfully propagates epistemic uncertainty in planning. 2. Planning in MCTS with an optimistic objective (Equation 11) is able to achieve deep exploration. 3. Planning can be leveraged for uncertainty estimation that improves over non-planning-based uncertainty estimation, even with learned dynamics models. We use BSUITE's [Osband et al., 2020] hard exploration benchmark Deep Sea of size 40 by 40. The Deep Sea environment encapsulates some of the hardest challenges associated with exploration: there is only one optimal action trajectory. The probability of finding the optimal trajectory through random action selection decays exponentially with the size of the environment. Every transition in the direction of the goal receives a negative reward that is negligible in comparison to the goal reward, but is otherwise the only reward the agent sees discouraging exploration in the direction that leads to the goal. Finally, the action mappings are randomized such that the effect of the same action is not the same in every state, preventing the agent from generalizing across actions. Three variations of the transition model f are investigated: (i) A **true** transition model (as in AlphaZero). The local uncertainty is estimated using RND. (ii) A value-equivalent (**abstracted**) transition model, where g, f are trained as in MuZero and a UBE loss. The local uncertainty is estimated using state-counts. (iii) An **anchored** transition model trained only to predict the true transition dynamics of the environment through a reconstruction

Table 1: Number of environment steps until the first visitation to the goal transition.

	Exploration	Average steps to goal transition for seeds that discovered goal ± STD	% seeds that discovered goal
True Model + RND	E-MCTS	$\mathbf{10539} \pm 9006$	94% of 35 seeds
	UBE	22801 ± 7514	91% of 35 seeds
	Uninformed	-	0% of 20 seeds
Abstracted Model + Counts	E-MCTS	$\mathbf{14339} \pm 6845$	100% of 23 seeds
	UBE	29945 ± 8113	57% of 21 seeds
	Uninformed	-	0% of 20 seeds
Anchored Model + RND	E-MCTS	$\mathbf{15241} \pm 3236$	95% of 20 seeds
	UBE	22497 ± 6645	85% of 20 seeds
	Uninformed	-	0% of 20 seeds

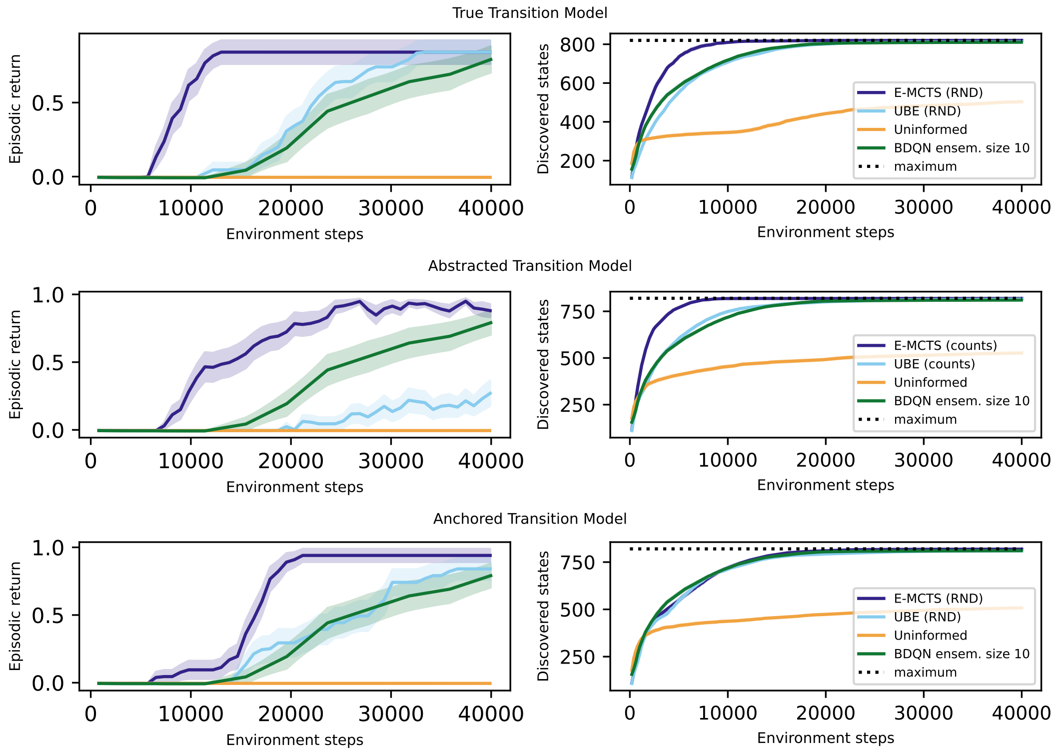

Figure 1: Mean and standard error for 20 seeds of different agents. Rows: Different transition models for the E-MCTS and UBE agents. Left: learning behavior presented as episodic return vs. environment steps. Right: exploration behavior presented as number of discovered states vs. environment steps.

loss L^k_{re}. The local uncertainty is estimated using RND. In all three transition model variations the reward r, value v and policy π functions are trained in the MuZero manner. Our implementation of the agents builds on the framework of MuZero as extended in EfficientZero Ye et al. [2021]. For implementation details see Appendices B.2 and B.4. For each model we compare four exploration methods: (i) **E-MCTS** (our method). (ii) An MCTS agent that plans without uncertainty and uses **UBE** for action selection over the predictions of the root (see Appendix B.7 for details). (iii) The MuZero exploration baseline which is **uninformed** with respect to epistemic uncertainty, relying on random action selection and Dirichlet noise. (iv) Bootstrapped DQN [**BDQN**, Osband et al., 2016] is a very strong model-free exploration approach on Deep Sea [Osband et al., 2020], and is included in the comparison for reference. The results are presented in Figures 1 and 2.

In Figure 1, E-MCTS demonstrates successful uncertainty propagation through successful deep exploration with all three transition model classes, supporting hypotheses 1 & 2. In addition, E-MCTS outperforms the UBE baseline in all 3 model classes, demonstrating improvement from planning with propagated uncertainty, supporting hypotheses 3. E-MCTS compares very well to the reference baseline BDQN, both in learning speed (Figure 1) as well as in scaling to larger environments (Figure 2, left), unlike the UBE baseline, which further points to performance gains from planning with epistemic uncertainty. Figure 3 shows how the estimated uncertainty about the value at the root node compares with ground-truth local uncertainty. Note that visited states that are close to terminal are associated with less epistemic root uncertainty, all states are explored, and the estimated uncertainty at the root node diminishes over training. Finally, we include an investigation of the sensitivity of E-MCTS to the exploration parameter β (Figure 2, right). Since exploitation and exploration episodes alternate, β need only be large enough to induce sufficient exploration to solve Deep Sea, resulting in low average regret across a wide range of values of β.

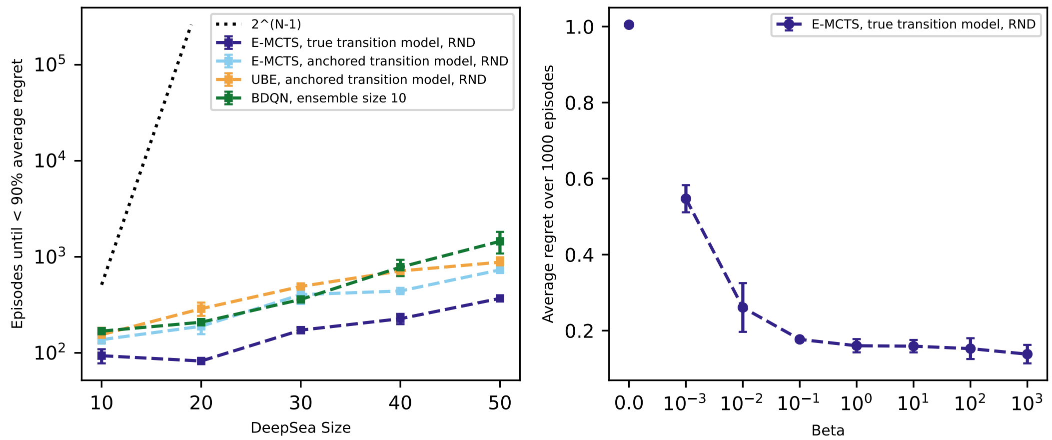

Figure 2: Left: Scaling to growing Deep Sea sizes [similar to Osband et al., 2020]. Mean of 5 seeds with standard error of the mean. Right: The effect of the exploration hyperparameter β, as average cumulative regret of 1000 episodes, for Deep Sea 30 by 30. Mean of 3 seeds with standard error.

6 Conclusions and Future Work

In this work we present E-MCTS, a novel method for incorporating epistemic uncertainty into MCTS. We use E-MCTS to modify the planning objective of MCTS to an exploratory objective to achieve deep exploration with MCTS-based MBRL agents. We evaluate E-MCTS on the Deep Sea benchmark, which is designed to be a hard exploration challenge, where our method yields significant improvements in state space exploration and uncertainty estimation. In addition, E-MCTS demonstrates the benefits of planning for exploration by empirically outperforming non-planning deep exploration baselines. The framework of E-MCTS provides a backbone for propagating uncertainty in other tree-based planning methods, as well as for the development of additional approaches to harnessing epistemic uncertainty. For example: (i) With E-MCTS, it is possible to plan with a conservative objective by discouraging uncertain decisions to improve reliability in the face of the unknown, which is paramount in the offline-RL setting. (ii) E-MCTS can be used to avoid planning into trajectories that increase epistemic uncertainty in value prediction, with the aim of achieving more reliable planning. (iii) Down-scaling of epistemically-uncertain targets has been used by Lee et al. [2021] and Wu et al. [2021] to improve the learning process of online and offline RL agents respectively. Given the advantages in exploration, it stands to reason that the improved value-uncertainty estimates from E-MCTS can benefit those approaches as well.

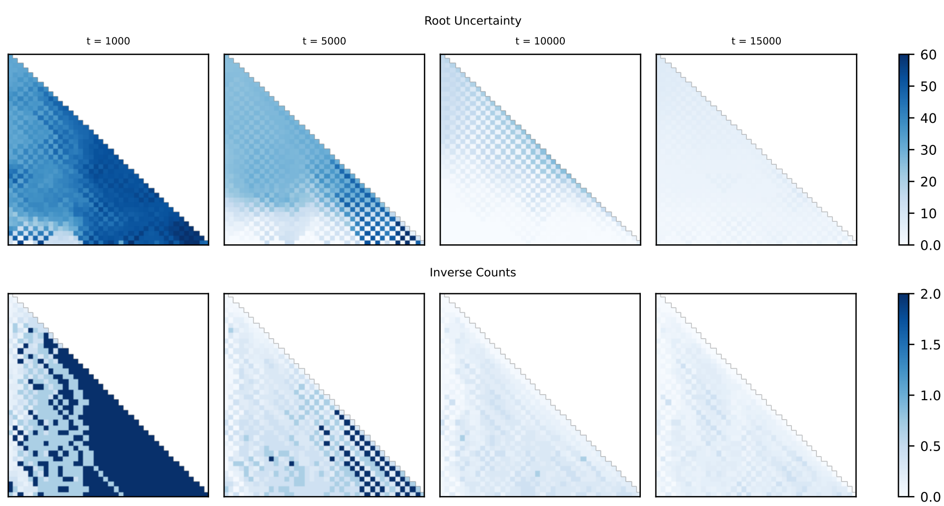

Figure 3: Heat maps of all states in the DeepSea environment of size 40 by 40 (lower triangle) at different times (columns) during an example training run of E-MCTS with true transition model and RND. Upper row: value uncertainty at the E-MCTS root node. Lower row: inverse visitation counts as reliable local uncertainty. Score of 2.0 represents unvisited.

Acknowledgements

We would like to thank Marco Loog, Frank van der Meulen, Itamar Sher, Moritz Zanger, Pascal van der Vaart & Joery de Vries for many fruitful discussions and helpful comments. We acknowledge the use of computational resources of the DelftBlue supercomputer, provided by Delft High Performance Computing Centre (https://www.tudelft.nl/dhpc) as well as the DAIC cluster. This work was partially supported by the EU Horizon 2020 programme under grant number 964505 (Epistemic AI).

References

Martial Agueh and Guillaume Carlier. Barycenters in the Wasserstein space. *SIAM Journal on Mathematical Analysis*, 43(2):904–924, 2011.

Ioannis Antonoglou, Julian Schrittwieser, Sherjil Ozair, Thomas K Hubert, and David Silver. Planning in stochastic environments with a learned model. In *International Conference on Learning Representations*, 2021.

Karl Johan Åström. Optimal control of Markov processes with incomplete state information I. *Journal of Mathematical Analysis and Applications*, 10:174–205, 1965.

Richard Bellman. A Markovian decision process. *Journal of mathematics and mechanics*, 6(5): 679–684, 1957.

Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Ivo Danihelka, Arthur Guez, Julian Schrittwieser, and David Silver. Policy improvement by planning with Gumbel. In *International Conference on Learning Representations*, 2022.

Jean-Bastien Grill, Florent Altché, Yunhao Tang, Thomas Hubert, Michal Valko, Ioannis Antonoglou, and Rémi Munos. Monte-Carlo tree search as regularized policy optimization. In *International Conference on Machine Learning*, pages 3769–3778. PMLR, 2020.

Mikael Henaff. Explicit explore-exploit algorithms in continuous state spaces. *Advances in Neural Information Processing Systems*, 32, 2019.

Todd Hester and Peter Stone. Intrinsically motivated model learning for a developing curious agent. In *2012 IEEE international conference on development and learning and epigenetic robotics (ICDL)*, pages 1–6. IEEE, 2012.

Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

David Janz, Jiri Hron, Przemysław Mazur, Katja Hofmann, José Miguel Hernández-Lobato, and Sebastian Tschiatschek. Successor uncertainties: exploration and uncertainty in temporal difference learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.

Nathan Lambert, Markus Wulfmeier, William Whitney, Arunkumar Byravan, Michael Bloesch, Vibhavari Dasagi, Tim Hertweck, and Martin Riedmiller. The challenges of exploration for offline reinforcement learning. *arXiv preprint arXiv:2201.11861*, 2022.

Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*, pages 6131–6141. PMLR, 2021.

Owen Lockwood and Mei Si. A review of uncertainty for deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 18, pages 155–162, 2022.

Amol Mandhane, Anton Zhernov, Maribeth Rauh, Chenjie Gu, Miaosen Wang, Flora Xue, Wendy Shang, Derek Pang, Rene Claus, Ching-Han Chiang, Cheng Chen, Jingning Han, Angie Chen, Daniel J. Mankowitz, Jackson Broshear, Julian Schrittwieser, Thomas Hubert, Oriol Vinyals, and Timothy A. Mann. MuZero with self-competition for rate control in VP9 video compression. *arXiv preprint arXiv:2202.06626*, 2022.

John Mern, Anil Yildiz, Zachary Sunberg, Tapan Mukerji, and Mykel J Kochenderfer. Bayesian optimized Monte-Carlo planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11880–11887, 2021.

Alberto Maria Metelli, Amarildo Likmeta, and Marcello Restelli. Propagating uncertainty in reinforcement learning via Wasserstein barycenters. *Advances in Neural Information Processing Systems*, 32, 2019.

Thomas M Moerland, Joost Broekens, Aske Plaat, and Catholijn M Jonker. The second type of uncertainty in Monte-Carlo tree search. *arXiv preprint arXiv:2005.09645*, 2020.

Thomas M Moerland, Joost Broekens, Aske Plaat, and Catholijn M Jonker. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118, 2023.

Brendan O'Donoghue, Ian Osband, Remi Munos, and Volodymyr Mnih. The uncertainty Bellman equation and exploration. In *International Conference on Machine Learning*, pages 3836–3845, 2018.

Ian Osband, Daniel Russo, and Benjamin Van Roy. (More) efficient reinforcement learning via posterior sampling. *Advances in Neural Information Processing Systems*, 26, 2013.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. *Advances in neural information processing systems*, 29:4026–4034, 2016.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvári, Satinder Singh, Benjamin Van Roy, Richard Sutton, David Silver, and Hado van Hasselt. Behaviour suite for reinforcement learning. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rygf-kSYwH`.

Aditya Ramesh, Louis Kirsch, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Exploring through random curiosity with general value functions. *Advances in Neural Information Processing Systems*, 35:18733–18748, 2022.

Christopher D Rosin. Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*, 61(3):203–230, 2011.

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, 2020.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pages 8583–8592. PMLR, 2020.

Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. Model-based active exploration. In *International conference on machine learning*, pages 5779–5788. PMLR, 2019.

David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. *Advances in neural information processing systems*, 23, 2010.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters Chess, Shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

Zachary N Sunberg and Mykel J Kochenderfer. Online algorithms for POMDPs with continuous state, action, and observation spaces. In *Twenty-Eighth International Conference on Automated Planning and Scheduling*, 2018.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Yue Wu, Shuangfei Zhai, Nitish Srivastava, Joshua M Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. Uncertainty weighted actor-critic for offline reinforcement learning. In *International Conference on Machine Learning*, pages 11319–11328. PMLR, 2021.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering Atari games with limited data. *Advances in Neural Information Processing Systems*, 34:25476–25488, 2021.

Sun Yi, F Gomez, and J Schmidhuber. Planning to be surprised: Optimal Bayesian exploration in dynamic environments. In *Proc. Fourth Conference on Artificial General Intelligence (AGI), Google, Mountain View, CA*, 2011.

A Law of Total Variance

The *law of total variance* for two continuous random variables X and Y can be derived as follows:

$$
\begin{aligned}
\mathbb{V}_Y[Y] &= \int (Y - \mathbb{E}_Y[Y])^2\, p(Y)\, dY \quad = \quad \iint (Y - \mathbb{E}_Y[Y])^2\, p(X,Y)\, dX\, dY \\
&= \iint (Y - \mathbb{E}_Y[Y])^2\, p(Y|X)\, p(X)\, dX\, dY \quad = \quad \mathbb{E}_X\Big[\mathbb{E}_Y\big[(Y - \mathbb{E}_Y[Y])^2 \big| X\big]\Big] \\
&= \mathbb{E}_X\Big[\mathbb{E}_Y\big[(Y - \mathbb{E}_Y[Y|X] + \mathbb{E}_Y[Y|X] - \mathbb{E}_Y[Y])^2 \big| X\big]\Big] \\
&= \mathbb{E}_X\Big[\underbrace{\mathbb{E}_Y\big[(Y - \mathbb{E}_Y[Y|X])^2 \big| X\big]}_{\mathbb{V}_Y[Y|X]}\Big] + 2\,\mathbb{E}_X\Big[\big(\underbrace{\mathbb{E}_Y[Y|X] - \mathbb{E}_Y[Y|X]}_{0}\big)\big(\mathbb{E}_Y[Y|X] - \mathbb{E}_Y[Y]\big)\Big] \\
&\quad + \underbrace{\mathbb{E}_X\Big[(\mathbb{E}_Y[Y|X] - \mathbb{E}_Y[Y])^2\Big]}_{\mathbb{V}_X[\mathbb{E}_Y[Y|X]]} \quad = \quad \mathbb{V}_X\big[\mathbb{E}_Y[Y|X]\big] + \mathbb{E}_X\big[\mathbb{V}_Y[Y|X]\big]
\end{aligned}
$$

B Implementation Details

B.1 Targets

In MuZero, the value targets v^{MCTS}_{t+k} for the prediction of value of latent state $\hat{s}^k_t$ that matches true state s_{t+k} are computed as an n-step TD target:

$$v^{\text{MCTS}}_{t+k} = \sum_{i=0}^{n-1} \gamma^i r_{t+k+i} + \gamma^n v^{\text{MCTS}}_{t+k+n}$$

Where v^{MCTS}_{t+k+n} can be computed in one of two ways:

(i) The value of the root of an MCTS tree computed for state s_{t+k+n}.

(ii) A prediction of the value network v for latent state $\hat{s}^0_{t+k+n}$.

Method (i) is expected to result in better value targets, but is more expensive computationally. Method (ii) is significantly cheaper computationally, but might hinder learning through the lack of value improvement (a max operator) on the value bootstrap. We refer to (i) as *root-based targets*.

The UBE target u^{target}_{t+k} for the prediction of value-uncertainty from the UBE head $u(\hat{s}^k_t)$ is computed a similar manner:

$$u^{\text{target}}_{t+k} = \sum_{i=0}^{n-1} \gamma^{2i} \sigma^2(\hat{s}^0_{t+k+i}, a_{t+k+i}) + \gamma^{2n} u_{t+k+n}$$

Analogous to the value target, the bootstrap u_{t+k+n} can be computed in two different ways:

(i) When E-MCTS is used, the target can be computed similarly to the MuZero value target, as the epistemic uncertainty of the root of an E-MCTS tree computed for state s_{t+k+n}. This tree can plan for an exploitatory objective (equation 1) to estimate the uncertainty of the value $V^\pi(s_{t+k+n})$, an exploratory objective (equation 11) to estimate the uncertainty of the value associated with the exploration policy, or even an uncertainty-maximizing objective:

$$a_k \quad := \quad \arg\max_{a_k} \sqrt{\sigma^2_q(\hat{s}_k, a_k)} + 2C\sqrt{\frac{2\log(\sum_{a'} N(\hat{s}_k, a'))}{N(\hat{s}_k, a_k)}}$$

Where the q term has been dropped entirely as an optimistic bound over the uncertainty to encourage exploration. Similarly, we refer to using as target the E-MCTS uncertainty prediction at the root as a root-based target. In our experiments, when UBE root-based target were used, we have used the uncertainty-maximizing objective.

(ii) When E-MCTS is not used, the UBE bootstrap u_{t+k+n} is computed as the maximum UBE over possible actions from state s_{t+k+n}:

$$u_{t+k+n} \;=\; \max_{a_{t+k+n}} \sigma^2(\hat{s}^0_{t+k+n}, a_{t+k+n}) + \gamma^2 u(f(\hat{s}^0_{t+k+n}, a_{t+k+n}))$$

These targets were used for all UBE-only agents, and for the E-MCTS agents that did not use root-based targets.

In all experiments we have used $n = 1$ (one-step targets) for the UBE targets.

In MuZero, the reward and value predictions $r(\hat{s}^k_t, a_{t+k}), v(\hat{s}^k_t)$ are represented as a discrete probability distribution over a range of discrete values $[-M, M]$, $M \in \mathbb{N}$. To transform the scalar value and reward targets to a categorical representation of the same representation format, a transformation function $\phi(x)$ is used, transforming a real number x into a categorical representation through a linear interpolation between its adjacent integers.

B.2 Losses

The original MuZero algorithm uses three loss functions:

$$\mathcal{L}_r \quad := \quad \tfrac{1}{|\mathcal{B}|} \sum_{t\in\mathcal{B}} \sum_{k=0}^{l-1} \phi(r_{t+k})^\top \log r(\hat{s}^k_t, a_{t+k})$$

$$\mathcal{L}_v \quad := \quad \tfrac{1}{|\mathcal{B}|} \sum_{t\in\mathcal{B}} \sum_{k=0}^{l-1} \phi(v^{\text{MCTS}}_{t+k})^\top \log v(\hat{s}^k_t)$$

$$\mathcal{L}_\pi \quad := \quad \tfrac{1}{|\mathcal{B}|} \sum_{t\in\mathcal{B}} \sum_{k=0}^{l-1} \pi^{\text{MCTS}}(s_{t+k})^\top \log \pi(\hat{s}^k_t)$$

Where $\mathcal{B} \equiv \{s_t, a_t, r_t, s_{t+1}, a_{t+1}, \dots, s_{t+l}\}_{t \in \mathcal{B}}$ is a training batch containing b trajectories of length l sampled from different episodes, r_{t+k} is the true reward observed in the environment, $r(\hat{s}_t^k, a_k), v(\hat{s}_t^k), \pi(\hat{s}_t^k)$ are respectively the reward value and policy predictions for latent state $\hat{s}_t^k$ (and action a_{t+k} when appropriate). $\pi^{\text{MCTS}}(s_{t+k})$ is a discrete probability distribution computed based on the normalized visitation counts to the children of an MCTS root computed at state s_{t+k} (see Equation 3).

In MuZero the gradient from the losses $\mathcal{L}_r, \mathcal{L}_v, \mathcal{L}_\pi$ propagates through the transition model f and are the only learning signal that is used to train the model. For the anchored model (see Section 5) we use an additional reconstruction loss:

$$\mathcal{L}_{re} := \frac{1}{|\mathcal{B}|} \sum_{t \in \mathcal{B}} \sum_{k=0}^{l-1} ||\hat{s}_t^k - s_{t+k}||^2$$

Which can alternatively be thought of as a consistency loss, where g is the identity function. The mean squared error loss is denoted with $\mathcal{L}_{\text{MSE}}$. To estimate value-uncertainty at the leaves, we train a UBE function u with a UBE loss $\mathcal{L}_u$:

$$\mathcal{L}_u := \frac{1}{|\mathcal{B}|} \sum_{t \in \mathcal{B}} \sum_{k=0}^{l-1} \phi(u_{t+k}^{\text{target}})^T \log \hat{u}_t^k$$

The final loss is computed as:

$$\mathcal{L} := \lambda_r \mathcal{L}_r + \lambda_v \mathcal{L}_v + \lambda_\pi \mathcal{L}_\pi + \lambda_u \mathcal{L}_u$$

Where the coefficients $\lambda_r, \lambda_v, \lambda_\pi, \lambda_u$ are used to weigh the relative effects the individual components of the loss have on the learned transition model f. When $\mathcal{L}_{re}$ was used (the anchored model in Section 5), the model parameters of f were affected only by L_{re}, through a second backwards pass.

B.3 Planning with Random Network Distillation Based Epistemic Uncertainty

We use RND to evaluate the transition uncertainty $\sigma^2(s_t, a_t)$ in planning with the true and anchored models. When the planning is done with a true model, the agent has access to the true states s_{t+k} and using RND to evaluate transition uncertainty over the state action pair (s_{t+k}, a_{t+k}) is natural. When the planning is done with the anchored model, the latent states outputted by the transition model $\hat{s}_t^k$ approximate the true states s_{t+k} which allows us to use RND over $(\hat{s}_t^k, a_{t+k})$. In both cases, RND is trained only over the observed transitions (s_{t+k}, a_{t+k}), not latent state representations $(\hat{s}_t^k, a_{t+k})$, to achieve the objective of yielding large RND prediction errors the further the latent state prediction $\hat{s}_t^k$ is from observed state s_{t+k}. As discussed in Section 3.4, we do not separate between state uncertainty and value / reward prediction uncertainty directly with RND and instead use $\mathbf{\Sigma}_k = 0$ and $\sigma_V(\hat{s}_k) = \max\Big(L_{rnd}(\hat{s}_{k-1}, a_{k-1}), u\big(f(\hat{s}_{k-1}, a_{k-1})\big)\Big)$.

B.4 Planning with Visitation-Counts Based Epistemic Uncertainty

When planning with the abstracted model, we provide the agent with access to two additional mechanisms that are used only for local uncertainty estimation: the true model $F(s_t, a_t)$ of the environment and a state-action visitation counter $C(s_t, a_t)$. During planning, the true transition model follows the planning decisions $a_{t:t+k}$ and keeps track of the true state s_{t+k}. When the agent evaluates the local uncertainty with transition $(\hat{s}_t^k, a_{t+k})$ the true model provides the matching true state s_{t+k} to the visitation counter, which produces the local uncertainty based on the following computation:

$$\sigma^2(s_{t+k}, a_{t+k}) = \frac{1}{C(s_{t+k}, a_{t+k}) + \epsilon}$$

Where $0 < \epsilon \leq 1$ is a constant and $C(s_{t+k}, a_{t+k})$ counts the number of times the state action pair (s_{t+k}, a_{t+k}) has been observed in the environment. This allows us to evaluate the abstracted-model agent in the presence of a reliable source of local uncertainty. The leaf-value uncertainty $u(\hat{s}_t^k)$ (which is the dominating factor in visited areas of the state space, as $\sigma^2(s_{t+k}, a_{t+k}) \to 0$ quickly in observed transitions) relies entirely on the learned UBE function u which operates directly on latent states $\hat{s}_t^k$.

B.5 Separating Exploration from Exploitation

Acting in the environment with a dedicated exploration policy can be expected to result in samples that are very off-exploitation-policy. Learning from very off-policy data is known for causing instability in training even in off-policy agents. To mitigate that, the E-MCTS and only-UBE agents (see section 5) alternate between two types of training episodes: *exploratory episodes* that follow an exploration policy throughout the episode (such as a policy generated by E-MCTS with an exploratory planning objective), and *exploitatory episodes* that follow the standard MuZero exploitation policy throughout the episode. This enables us to provide the agent with quality exploitation targets to evaluate and train the value and policy functions reliably, while also providing a large amount of exploratory samples that explore the environment much more effectively and are more likely to efficiently search for high-reward interactions.

In practice, rather than alternate between exploration and exploitation episodes we run a certain number of episodes in parallel, a certain portion of which are exploitatory and the rest are exploratory. In our experiments the ratio was $50/50$. During exploration episodes, we do not wish to bias the search in the tree with respect to previously tried actions, but rather only with respect to the combination of value and uncertainty (equation 11). We set the policy prediction $\pi(\hat{s}_t^k)$ (see Equation 2) to uniform over all actions, for all $\hat{s}_t^k$ during exploration episodes. In addition, Dirichlet noise was not used to drive exploration in MCTS with the UBE and E-MCTS agents.

B.6 Environment Adaptation

To maintain the exploration difficulty of Deep Sea while reducing numerical challenges, we amplify the goal reward from 1 to 10. To limit the challenge of learning a model that can distinguish between approximately N^2 unique states when learning the true dynamics of the environment, while retaining the exploration challenge of searching for one trajectory in a total of 2^N trajectories, we choose environment size $N = 40$, for a $(40, 40)$ grid. To further simplify model learning with the anchored model, the representation function g that was used for the anchored model transforms the observations from 2 dimensional (N, N) one-hot representations to 1 dimensional $(2N)$ representations where the first N entries are a 1-hot vector representing the row and following N entries are a 1-hot vector representing the column. From this perspective, we can view the $\mathcal{L}_{re}$ loss that was used to train the anchored model as a consistency loss between the representation and the state prediction rather than a reconstruction loss. The loss itself is the same loss specified in Appendix B.2.

B.7 UBE Baseline

The UBE baseline agent uses MCTS to evaluate the value of actions using MCTS in the same manner as Alpha/MuZero, and explores by taking the action a_t that maximizes the combination of the Q-values approximated by MCTS q, local uncertainty σ^2 and UBE u:

$$a_t = \arg\max_a q(\hat{s}_0, a_t) + \beta\sqrt{\sigma^2(\hat{s}_0, a_t) + \gamma^2 u(f(\hat{s}_0, a_t))}. \tag{14}$$

B.8 Compute

The experiments were run on the Delft Blue and DAIC computation clusters, using any of the following GPU architectures: NVIDIA Quadro K2200, Tesla P100, GeForce GTX 1080 Ti, GeForce RTX 2080 Ti, Tesla V100S and Nvidia A-40. Each seed was ran on one GPU, and was given access to 100 GB of RAM and 16 CPU cores. Total training time was in the range of 12 to 65 hours per seed, depending on GPU architecture and whether root-based targets (see Appendix B.1) which significantly increased training time were used or not.

C Network Architecture & Hyperparameters

C.1 Hyperparameter Search

Due to the large number of hyperparameters in the MuZero framework, our optimization process consisted of manual modifications to the hyperparameters used by Ye et al. [2021] with the objective of achieving learning stability on the target environment with the simplest network architectures

Table 2: Network architecture hyperparameters

	True Model	
Function	Hidden Layers Sizes	Output Layer Size
f	-	-
g	-	-
r	[256, 256]	21
v	[256, 256]	21
u	[256, 256]	21
π	[256, 256]	2
	Anchored Model	
Function	Hidden Layers Sizes	Output Layer Size
f	[1024, 1024, 1024]	80
g	-	-
r	[256, 256]	21
v	[256, 256]	21
u	[256, 256]	21
π	[256, 256]	2
	Abstracted Model	
Function	Hidden Layers Sizes	Output Layer Size
f	[1024, 1024, 1024]	100
g	[512, 512]	100
r	[128, 128]	21
v	[128, 128]	21
u	[128, 128, 128]	21
π	[128, 128]	2
	RND network architecture	
Function	Hidden Layers Sizes	Output Layer Size
ψ	[1024, 1024]	512
ψ'	[512]	512

possible. Two exceptions to this statement are the RND network architecture and scale, and the exploration parameter β.

The RND architecture was designed with the objective of reliably achieving small RND predictions over observed state-action pairs and large predictions over unobserved state-action pairs. The RND scale was tuned with the objective of achieving local uncertainty measures for unobserved state-action pairs that are significantly larger than the minimum reward of Deep Sea.

The β parameter was tuned with the objective that the E-MCTS and only-UBE agents will prioritize exploration of the environment over exploitation until the entire environment has been searched, and was tuned separately for every model.

C.2 Network Architecture

The functions $f, g, r, v, u, \pi, \psi, \psi'$ used fully connected DNNs of varying sizes. The sizes of the hidden layers and output layers are specified in Table 2.

C.3 Hyperparameter Configuration

We detail the full set of hyperparameters in Tables 3 and 4. For the BDQN baseline, we used the default implementation in `https://github.com/deepmind/bsuite`, with ensemble size of 10 and matching batch size to E-MCTS: number of unroll steps times batch size $5 \cdot 256 = 1230$. Otherwise, the default hyper parameters were used.

Table 3: Shared across all models and agents

Parameter	Setting	Comment
Stacked Observations	1	
γ	0.995	
Number of simulations in MCTS	50	
Dirichlet noise ratio (ξ)	0.3	
Root exploration fraction	0	
Batch size	256	
Learning rate	0.0005	
Optimizer	Adam [Kingma and Ba, 2015]	
Unroll steps l	5	
Value target TD steps (n_v)	5	
UBE target TD steps (n_u)	1	
value support size	21	
UBE support size	21	
Reward support size	21	
Reanalyzed policy ratio	0.99	See [Ye et al., 2021]
Prioritized sampling from the replay	True	See [Schrittwieser et al., 2020] Appendix G
Priority exponent (α)	0.6	See [Schrittwieser et al., 2020] Appendix G
Priority correction (β_p)	$0.4 \to 1$	See [Schrittwieser et al., 2020] Appendix G
Evaluation episodes	8	
Min replay size for sampling	300	
Self-play network updating inerval	5	
Target network updating interval	10	

Table 4: Specific for models and agents

Parameter	Setting								
	True Model			Abstracted Model			Anchored Model		
	E-MCTS	UBE	Uninf.	E-MCTS	UBE	Uninf.	E-MCTS	UBE	Uninf.
Training steps / environment interactions	45K	45K	45K	35K	35K	35K	45K	45K	45K
Reward loss weight λ_r	1	1	1	1	1	1	1	1	1
Value-loss weight λ_v	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Policy-loss weight λ_π	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
UBE-loss weight λ_u	0.125	0.125	-	0.25	0.25	-	0.125	0.125	-
RND scale	1.0	1.0	-	-	-	-	0.001	0.001	-
Root based targets	False	False	False	True	True	True	False	False	False
Disabled policy in exploration	True	True	False	True	True	False	True	True	False
Number of parallel episodes	2	2	2	2	2	2	2	2	2
Out of are exploration episodes	1	1	-	1	1	-	1	1	-
Exploration coefficient β	10	10	-	1	1	-	10	10	-
Dirichlet noise magnitude ρ	0	0	0.25	0	0	0.25	0	0	0.25