Epistemic Monte Carlo Tree Search

Yaniv Oren
Delft University of Technology
2628 CD Delft, The Netherlands
y.oren@tudelft.nl

Viliam Vadocz
Delft University of Technology
2628 CD Delft, The Netherlands

Matthijs T. J. Spaan
Delft University of Technology
2628 CD Delft, The Netherlands
m.t.j.spaan@tudelft.nl

Wendelin Böhmer
Delft University of Technology
2628 CD Delft, The Netherlands
j.w.bohmer@tudelft.nl

Abstract

The AlphaZero/MuZero (A/MZ) family of algorithms has achieved remarkable success across various challenging domains by integrating Monte Carlo Tree Search (MCTS) with learned models. Learned models introduce epistemic uncertainty, which is caused by learning from limited data and is useful for exploration in sparse reward environments. MCTS does not account for the propagation of this uncertainty however. To address this, we introduce Epistemic MCTS (EMCTS): a theoretically motivated approach to account for the epistemic uncertainty in search and harness the search for deep exploration. In the challenging sparse-reward task of writing code in the Assembly language SUBLEQ, AZ paired with our method achieves significantly higher sample efficiency over baseline AZ. Search with EMCTS solves variations of the commonly used hard-exploration benchmark Deep Sea - which baseline A/MZ are practically unable to solve - much faster than an otherwise equivalent method that does not use search for uncertainty estimation, demonstrating significant benefits from search for epistemic uncertainty estimation.

1 Introduction

Many recent successes of reinforcement learning (RL) have been achieved by the model-based algorithm family of AlphaZero/MuZero (A/MZ, Silver et al., 2018; Schrittwieser et al., 2020). A/MZ have outperformed humans in games, in tasks that traditionally relied on intricate human engineering (Mandhane et al., 2022) and even made real world impact with the design of novel, more efficient algorithms (Fawzi et al., 2022; Mankowitz et al., 2023) for day-to-day problems, a task that is often formulated as a challenging sparse reward environment. When rewards are sparse, it is difficult to learn good policies without employing some form of *deep exploration* to search for the rewards. Deep exploration refers to the ability of the agent to direct itself towards novel transitions in the environment irrespective of how far away they are from the current state and promises up to exponential improvement in sample efficiency in sparse reward environments (Osband et al., 2018).

At the core of A/MZ is the combination of Monte Carlo Tree Search (MCTS, Swiechowski et al., 2023) with learned models of value and/or environment dynamics. Learned models introduce *epistemic uncertainty*, which refers to the uncertainty in the predictions of the model sourced in limited coverage of the state-action space during training (Hüllermeier & Waegeman, 2021). Accounting for epistemic uncertainty allows the agent to discern between predictions that are based on evidence (i.e. the learned predictor was trained on this input), or based on generalization (i.e. the learned predictor *was not* trained on this input) and is useful for many purposes in online and offline RL. Common uses range from reducing overestimation errors through pessimism in the face of uncertainty (see Kumar et al., 2020), to directing deep exploration through optimism in the face of uncertainty (see

Neu & Pike-Burke, 2020). Harnessing search for exploration is a popular approach in similar model based algorithms, such as Dreamer (Sekar et al., 2020).

MCTS, however, was designed for search with the true dynamics model and without a value model, and as a result, does not account for epistemic uncertainty introduced from learning the models. For this reason, A/MZ cannot harness MCTS for deep exploration, nor benefit from the epistemic uncertainty associated with the predictions of the search tree in other ways. In this work we aim to address both, with three main contributions. Practical and theoretically motivated methods to (i) harness MCTS and epistemic uncertainty for upper-confidence-bound-based deep exploration (Jin et al., 2018) and (ii) propagate the epistemic uncertainty from learned models of value and/or dynamics during search, which we call Epistemic MCTS (EMCTS). (iii) A parallelized implementation in JAX (Bradbury et al., 2018) of EMCTS paired with an AZ agent and an environment implementing the Assembly language SUBLEQ (Mazonka & Kolodin, 2011)[1]. We find that the propagation of epistemic uncertainty in EMCTS is very similar to that of value in MCTS. As search with MCTS improves the value estimates at the root, we hypothesize that search with EMCTS similarly improves the epistemic uncertainty estimates at the root, resulting in more accurate UCBs and more sample efficient exploration compared to a method that does not rely on search but is otherwise equivalent.

We evaluate EMCTS in the challenging, similar to real-world applications and sparse-reward task of programming in SUBLEQ, as well as in the commonly used hard-exploration benchmark Deep Sea (Osband et al., 2020). Our method is able to find correct programs for a harder programming task in a much smaller number of samples than the AZ baseline. In the Deep Sea benchmark, our method demonstrates deep exploration by solving stochastic and deterministic reward variations of the task, both of which baseline A/MZ is unable to solve in a reasonable number of samples. In addition, EMCTS significantly outperforms an ablation that does not rely on search for epistemic uncertainty estimation but is otherwise equivalent, demonstrating significant advantages from search for uncertainty estimation.

2 Background

In RL, an agent learns a behavior policy through interactions with an environment. The environment is represented by a Markov Decision Process (MDP, Bellman, 1957) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \rho, \mathcal{R}, P, \gamma \rangle$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ a set of actions, ρ the initial state distribution, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ a bounded possibly stochastic reward function, and P is a transition distribution such that $P(s'|s,a)$ specifies the probability of transitioning from state s to state s' after executing action a. The objective J_π of the agent is to find a policy $\pi(a|s)$, specifying the probability of selecting action a in state s, that maximizes the *expected discounted return*, also called value V^π, from the starting state distribution ρ:

$$J_\pi = \mathbb{E}[V^\pi(s_0)|\, s_0 \sim \rho\,] = \mathbb{E}\Big[\sum_{t=0}^{H-1} \gamma^t \mathcal{R}(s_t, a_t) \Big|\, s_0 \sim \rho, s_{t+1} \sim P(s_t, a_t), a_t \sim \pi(s_t)\Big]. \tag{1}$$

The discount factor $0 < \gamma < 1$ is used in infinite-horizon MDPs, i.e. $H = \infty$, to guarantee that the values remain bounded, and is commonly used in RL for training stability. A state-action *Q-value function* is also often used: $Q^\pi(s,a) = \mathbb{E}[\mathcal{R}(s,a) + \gamma V^\pi(s')|\, s' \sim P(s,a)]$. We denote the value of the optimal policy π^* with $V^*(s) = \max_\pi V^\pi(s), \forall s \in \mathcal{S}$. In offline RL, the agent must maximize this objective given a static dataset. In model-based RL (MBRL) the agent uses a model of the dynamics of the environment $(P, \mathcal{R})$ to optimize its policy, often through planning (Moerland et al., 2023). The dynamics are either learned from interactions (e.g., in MZ, Schrittwieser et al., 2020) or provided (e.g., in AZ, Silver et al., 2018). In Deep MBRL the agent utilizes deep neural networks (DNN, Goodfellow et al., 2016) to approximate any of the value, policy, reward and transition functions.

[1]Our implementation, inspired by SIC-1 which is an open source game demonstrating SUBLEQ, is available at `https://github.com/emcts/e-alphazero`.

2.1 Monte Carlo Tree Search

The MCTS algorithm constructs a tree with the current state s_t at its root to estimate the objective: $\arg\max_a \max_\pi Q^\pi(s_t, a)$ (Browne et al., 2012), by iteratively performing selection, expansion, simulation and backup. At each iteration i of the algorithm a trajectory in the tree is selected using a tree search policy such as UCT (Kocsis & Szepesvári, 2006):

$$a_k^i \;=\; UCT_i(s_k) \;=\; \arg\max_{a \in A} q^i(s_k, a) + C_{UCT}\sqrt{\tfrac{2\log(\sum_{a'} N(s_k, a'))}{N(s_k, a)}}, \tag{2}$$

where $N(s_k, a)$ denotes the number of times action a has been selected in node s_k, $C_{UCT} > 0$ trades off exploration of new nodes in the tree with maximizing observed return and $q^i(s_k, a)$ is the averaged return observed for this state action up to step i. Modern algorithms (such as A/MZ) use instead variations of PUCT (Rosin, 2011):

$$a_k^i \;=\; PUCT_i(s_k) \;=\; \arg\max_{a \in A} q^i(s_k, a) + \pi(a|s_k)\, C_{PUCT}\tfrac{\sqrt{\sum_{a'} N(s_k, a')}}{1+N(s_k, a)}, \tag{3}$$

with some learned prior-policy π. When selection step i arrives at a leaf s_T^i the node is expanded with a value estimate $v(s_T^i)$. MCTS propagates the return ν^i of planning step i back along the planning trajectory $\{s_j^i, a_j^i\}_{j=0}^T$:

$$\nu^i(s_k, a_k^i) = \sum_{j=0}^{T-1} \gamma^j r(s_{k+j}^i, a_{k+j}^i) + \gamma^T v(s_T^i), \quad q^i(s_k, a) = \frac{1}{N(s_k, a)} \sum_{j=1}^{i} \nu^j(s_k, a) \tag{4}$$

where $a_{k+j}^i = P/UCT_i(s_{k+j}^i)$ and $s_{k+j+1}^i = f(s_{k+j}^i, a_{k+j}^i)$ for a deterministic transition function[2] $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ and a mean-reward function $r(s, a) = \mathbb{E}[\mathcal{R}(s, a)]$. In the original MCTS, the dynamics model $m = (f, r)$ is provided to the agent and assumed to be correct for the true MDP $\mathcal{M}$ and the value estimate $v(s_T^i)$ is computed using Monte-Carlo (MC) rollouts with m. In AZ the value function (Silver et al., 2018) and in MZ all functions f, r, v, are learned from interactions with the environment and thus their predictions are uncertain outside of the training set. The prior-policy π is trained with cross-entropy loss on targets extracted from the root of the tree. With Reanalyze (Schrittwieser et al., 2021), which uses search to generate fresh value and policy targets from off-policy data, A/MZ are able to learn off-policy.

2.2 Quantifying Uncertainty in Deep Reinforcement Learning

Quantifying uncertainty in deep learning is an active field of research (see Hüllermeier & Waegeman, 2021; Lockwood & Si, 2022). In this work, we take the common approach for quantifying epistemic uncertainty as *the variance in a random variable that approximates predictions that are consistent with a dataset of observed interactions* $\mathcal{D}$. We model a learned function $\hat{r}$, trained to approximate a true function r on data $\mathcal{D}$, $\hat{r}(s, a) \approx r(s, a)$, as a random variable with respect to the data: $\hat{r}(s, a)|\mathcal{D} := \hat{R}(s, a)$. We assume unbiased approximation $\mathbb{E}[\hat{R}(s, a)] := r(s, a)$, and use the variance $\mathbb{V}[\hat{R}(s, a)]$ to quantify the epistemic uncertainty. In discrete state-action spaces, the epistemic uncertainty in a reward or a transition can be estimated as the novelty of a state or state-action pair using visitation counting. Exact counting, which is generally intractable in large state-action spaces, can be replaced by Hash-based counting (Tang et al., 2017), or methods that focus on direct estimation of the novelty of state-action pairs, such Random Network Distillation (RND, Burda et al., 2019).

In contrast, epistemic uncertainty in a *value* prediction $\mathbb{V}[\hat{V}(s)]$ that is trained with TD-based targets, requires reasoning about uncertainty in the value-targets, as well as whether $\hat{V}$ was trained on s at all. We follow the popular approach by Strens (2000) of defining epistemic uncertainty in the value function as the variance of a Bayesian posterior of the Q-values of a

[2]For notational simplicity we assume here a deterministic transition function f, so that nodes s_{k+j}^i correspond to individual states. It is also possible to use a stochastic transition model P, where nodes correspond to distributions of states, which are sampled by the selection step. The above equations remain the same, but the state corresponding to s_{k+j}^i is effectively a random variable.

policy conditioned on the data the agent has collected, as follows:

$$\mathbb{V}[\hat{Q}^\pi(s,a)] = \mathbb{V}\Big[\hat{R}(s,a) + \gamma \sum_{s',a'} \pi(a'|s')\, \hat{P}(s'|s,a)\, \hat{Q}^\pi(s',a')\Big]. \tag{5}$$

Note that the epistemic uncertainty about rewards $\mathbb{V}[\hat{R}]$ and transitions $\mathbb{V}[\hat{P}]$ are induced by the data $\mathcal{D}$ the value function has been trained on, not by the agent's knowledge of the environment. Values predicted by learned value functions can therefore be epistemically uncertain, even if a perfect model of the environment is known to the agent. In their work on the Uncertainty Bellman Equation (UBE), O'Donoghue et al. (2018) propose to approximate an upper bound $u^\pi(s_t,a_t) \geq \mathbb{V}[\hat{Q}^\pi(s_t,a_t)]$. $u^\pi(s_t,a_t)$ can be learned with (possibly n-step) TD targets in a similar manner to value learning from *local* uncertainties $\eta(s,a)$:

$$u^\pi(s_t,a_t) := \eta(s_t,a_t) + \gamma^2 \textstyle\sum_{a'} \pi(a'|s_{t+1})\, u^\pi(s_{t+1},a') \leq \eta(s_t,a_t) + \gamma^2 \max_{a'} u^\pi(s_{t+1},a'). \tag{6}$$

The local uncertainty η can be derived from $\mathbb{V}[\hat{R}]$ and $\mathbb{V}[\hat{P}]$. Note that the above inequality yields an upper bound on the epistemic uncertainty of *any* policy π with training data $\mathcal{D}$.

2.3 Deep Exploration with Upper Confidence Bounds

A popular approach to harness epistemic uncertainty for deep exploration is that of optimism in the face of uncertainty (Lattimore & Szepesvari, 2017), often formalized using an upper confidence bound (UCB) on the true value of the optimal policy (Azar et al., 2017; Jin et al., 2018). Acting greedily with respect to the maximum UCB guarantees exploration of the environment, as long as the UCB tightens with more samples from the environment. The efficiency of UCB exploration depends on the epistemic uncertainty estimator. In environments with small state-action spaces, epistemic uncertainty estimators that predict maximum uncertainty for unvisited states can be used to guarantee exploration of every state-action pair. To explore continuous or large state-action domains in practical numbers of samples, using an UCB requires uncertainty estimators that estimate the epistemic uncertainty in unvisited states s' based on a similarity metric between s' and visited states $s \in \mathcal{D}$ (Jin et al., 2023), such as imposed by RND, for example.

3 Deep Exploration with Epistemic MCTS

To find an optimal policy, an RL agent must explore the environment until all necessary information is gathered. When the agent uses search for action selection in the environment, it can search for exploration, exploitation, or a trade-off between the two. When a learned model $\hat{m}$ is used in search, the agent faces a problem: the values $q^i_{\hat{m}}$ in the search tree converge to $Q^{*\hat{m}}_{\hat{m}}$, the Q value *in the learned model* of the policy that is optimal *in that model* $\pi^*_{\hat{m}}$. In areas where the learned model is inaccurate, the search may lead to arbitrarily bad actions with respect to expected return in the true environment. From the perspective of exploration on the other hand, this presents an opportunity: by estimating the epistemic uncertainty the agent can identify the areas where the model is uncertain due to insufficient interactions, and use the uncertainty to direct exploration into these areas.

Our objective is then two-fold: To extend MCTS to estimate and propagate the epistemic uncertainty from the uncertain learned model and harness the epistemic uncertainty in the search to achieve deep exploration of the environment. To achieve this, we take the following steps: (i) Formulate the learned model $\hat{m}$ as a random variable and use it to construct a UCB on Q^* (Section 3.1). (ii) Propose search policies to track the maximum UCB exploration objective (Section 3.2). (iii) Propagate the epistemic uncertainty through search, such that the epistemic uncertainty $\mathbb{V}[q^i_{\hat{m}}(s,a)]$ in nodes' value predictions $q^i_{\hat{m}}(s,a)$ can be estimated (Section 3.3). Search with learned transition models introduces additional challenges for uncertainty propagation, which we address in Section 3.4.

3.1 Search with a Learned Reward Model

For simplicity, we begin by formulating the problem of search with learned models only in terms of a learned reward model and later extend the setup to learned value

and transition models. Consider the following setting: the agent has access to a dataset $\mathcal{D} = \{(s_i, a_i, r_i, s_{i+1}) \,|\, 0 \leq i < N\}$ of transitions and rewards, as well as to the true (possibly stochastic) transition model P. The reward function is bounded with: $\max_{s,a \in \mathcal{S} \times A} |\mathcal{R}(s,a)| \leq r_{max}$. We define the uncertain model as a random variable $\hat{M} = (P, \hat{R})$. The uncertain mean-reward function is defined as a random variable $\hat{R}$ in the standard manner for defining an epistemically uncertain model (see Section 2.2), such that $\mathbb{E}[\hat{R}(s,a)] = r(s,a)$ the mean reward, and $\hat{R}(s,a) = \frac{1}{|(s,a,\cdot,\cdot) \in D|} \sum_{(s,a,r,\cdot) \in D} r \approx r(s,a)$, as the empirical mean of observed rewards. The epistemic uncertainty in reward prediction is defined as the variance $\mathbb{V}[\hat{R}(s,a)]$. We note that $\hat{R}(s,a)$ is a bounded random variable over the interval $[-r_{max}, r_{max}]$. To facilitate constructing an upper confidence bound on Q^* we define $\mathbb{V}[\hat{R}(s,a)] := r_{max}^2, \forall (s,a) \notin \mathcal{D}$. That is, for unobserved transitions, the maximum variance possible for a bounded random variable (Popoviciu, 1935). The value with respect to the Markov chain induced by the random variable $\hat{M}$ is then itself a random variable:

$$Q^{\pi}_{\hat{M}}(s,a) := \mathbb{E}_{\pi,P}\Big[\sum_{i=0}^{\infty} \gamma^i r_i \,\Big|\, a_i \sim \pi(s_i), s_{i+1} \sim P(s_i, a_i), r_i = \hat{R}(s_i, a_i), s_0 = s, a_0 = a \Big], \quad (7)$$

where the expectation is with respect to π and P and not $\hat{R}$. Using this definition for the value in the model $Q^{\pi}_{\hat{M}}$ and its variance $\mathbb{V}[Q^{\pi}_{\hat{M}}]$ we can construct an upper confidence bound on Q^* for deep exploration.

Theorem 1 *For $\hat{M}$, $\mathcal{M}$, Q^*, $Q^{\pi}_{\hat{M}}$ defined as above and $\delta \in (0, 1]$:*

$$P\left(Q^*(s,a) \leq \max_{\pi} \left(Q^{\pi}_{\hat{M}}(s,a) + \tfrac{1}{\sqrt{\delta}} \sqrt{\mathbb{V}[Q^{\pi}_{\hat{M}}(s,a)]} \right) \right) \;\geq\; 1 - \delta\,, \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}. \quad (8)$$

The proof is provided in Appendix A.1 and relies on the linearity of $Q^{\pi}_{\hat{M}}$ in the reward function $\hat{R}$. An extended UCB that is maintained in the presence of an uncertain transition model $\hat{P}$ is constructed in Appendix A.3. The UCB can be tracked by MCTS using epistemic search policies, which we propose next.

3.2 Planning for Exploration with Epistemic Search

To harness search for deep exploration through the approximation of the UCB in Equation 8, we propose Epistemic P/UCT (EP/UCT, changes are marked in blue):

$$a_{\text{EUCT}} \;=\; \arg\max_{a \in A} q^{\beta}_{\hat{M}}(s_k, a) + C_{\text{UCT}} \sqrt{\tfrac{2 \log(\sum_{a'} N(s_k, a'))}{N(s_k, a)}}, \quad (9)$$

$$a_{\text{EPUCT}} \;=\; \arg\max_{a \in A} q^{\beta}_{\hat{M}}(s_k, a) + \pi(a|s_k)\, C_{\text{PUCT}} \tfrac{\sqrt{\sum_{a'} N(s_k, a')}}{1 + N(s_k, a)}, \quad (10)$$

where we define:

$$q^{\beta}_{\hat{M}}(s_k, a) := \underbrace{\tfrac{1}{N(s_k,a)} \textstyle\sum_{i=0}^{N(s_k,a)} \nu^i_{\hat{M}}(s_k, a)}_{:= q_{\hat{M}}(s_k, a)} + \beta \underbrace{\tfrac{1}{N(s_k,a)} \textstyle\sum_{i=0}^{N(s_k,a)} \sqrt{\mathbb{V}[\nu^i_{\hat{M}}(s_k, a)]}}_{:= \sigma_{q_{\hat{M}}}(s_k,a) \geq \sqrt{\mathbb{V}[q_{\hat{M}}(s_k,a)]}}. \quad (11)$$

The hyper-parameter $\beta \geq 0$ can be tuned per task, or chosen to guarantee an upper confidence bound with specific confidence $1 - \delta$, and $\nu^i_{\hat{M}}(s_k, a)$ is as defined in Equation 4 for a specific model $\hat{M}$. We suppress the dependence on planning step i in the notation of $q^{\beta}_{\hat{M}}(s_k, a)$ for simplicity. To maintain the property of PUCT, which assumes $q_{\hat{M}}$ is between 0 and 1 (Rosin, 2011), one can use the Q normalization approach proposed by Schrittwieser et al. (2020) to normalize the q^{β} scores. More modern variations of PUCT such as those proposed by Danihelka et al. (2022) can be used by replacing $q_{\hat{M}}$ with $q^{\beta}_{\hat{M}}$ in the search objective. We note that in order to properly harness search policies that use a prior-policy for deep exploration, such as EPUCT, the agent must learn an exploration-prior-policy π_e that is trained with an exploratory objective. Otherwise, the prior policy may direct the search away from the exploratory planning objective. To enable EP/UCT, in the following section

we propose methodology to estimate the epistemic uncertainty in backups and upper bound $\mathbb{V}[q_{\hat{M}}(s_k, a)]$.

3.3 Propagating Epistemic Uncertainty in Search

To estimate $\mathbb{V}[q_{\hat{M}}(s_k, a)]$ we need to: (i) extend the problem setup to account for the learned value model, (ii) compute or upper bound the epistemic uncertainty in one backup $\mathbb{V}[\nu^i_{\hat{M}}(s_k, a)]$ and (iii) compute or upper bound $\mathbb{V}[q_{\hat{M}}(s_k, a)]$ using $\mathbb{V}[\nu^i_{\hat{M}}(s_k, a)]$.

Search with a Learned Value Model A learned value model $\hat{V}$ introduces an additional source of epistemic uncertainty into the search, $\mathbb{V}[\hat{V}(s)]$. In this case, for the UCB to hold it is important that the models $\hat{M} = (f, \hat{R}, \hat{V})$ are trained on the same data, which is the popular choice in practice. This is sufficient to analytically maintain that $\mathbb{V}[\hat{V}(s)] \geq \mathbb{V}[V^\pi_{\hat{M}}(s)]$ (see Equation 5). To maintain this property in practice, we choose the UBE (O'Donoghue et al., 2018) estimator $\hat{u}$ (see Section 2.2 and Equation 6) which upper bounds $\mathbb{V}[\hat{V}(s)]$ using $\mathbb{V}[\hat{R}(s, a)]$. To account for the possibility that $\hat{u}$ is unreliable outside of the training set we use the novelty of (s, a) to upper bound the uncertainty predicted by $\hat{u}$ (see Appendix D.6 and Equation 25). When the reward model is not learned, such as in AZ, the value model still has epistemic uncertainty as it is trained from interactions and $\mathbb{V}[\hat{V}(s)]$ coincides with the definition of value uncertainty in model-free literature (Equation 5). This definition accounts for uncertainty in transitions that are unobserved in the environment, regardless of whether the transitions are known to the planning model, capturing the uncertainty in a value model that is trained from interactions. To estimate $\mathbb{V}[\hat{R}(s, a)]$ RND and (pseudo-)counting methods can be used (see Section 2.2 for more detail).

Epistemic Uncertainty of the Backup The epistemic uncertainty in one backup step $\mathbb{V}[\nu^i_{\hat{M}}(s_k, a)]$ starting at node s_k and choosing action a can be formulated as follows:

$$\mathbb{V}[\nu^i_{\hat{M}}(s_k, a)] = \sum_{j=0}^{T-k-1} \gamma^{2j} \mathbb{V}[\hat{R}(s^i_{k+j}, a^i_{k+j})] + \gamma^{2(T-k)} \mathbb{V}[\hat{V}(s^i_T)], \tag{12}$$

under the assumption that $\hat{R}(s_{k+i}, a)$ is independent from $\hat{R}(s_{k+j}, a), \forall i \neq j$, and $\hat{V}$ is independent from $\hat{R}$ conditional on the data. We note that this assumption might be reasonable for some estimators (i.e., a table), but is not generally assumed to be true for DNNs. To avoid this assumption, we can instead upper bound the standard deviation of correlated backups using the sum of standard deviations (see Appendix A.2).

Epistemic Uncertainty of Node Values Since A/MZ use the same model $\hat{M}$ throughout planning, the returns predicted for different backups cannot be assumed to be de-correlated. We propose to upper bound the epistemic uncertainty instead:

$$\mathbb{V}[q_{\hat{M}}(s_k, a)] \leq \sigma^2_{q_{\hat{M}}}(s_k, a) := \left(\frac{1}{N(s_k, a)} \sum_{i=0}^{N(s_k,a)} \sqrt{\mathbb{V}[\nu^i(s_k, a)]} \right)^2, \tag{13}$$

where $N(s_k, a)$ is the number of visitations to action a at node s_k. We provide a complete derivation in Appendix A.2. Equation 13 completes the Epistemic MCTS (EMCTS) algorithm for learned value and/or reward models. See Algorithm 1 for pseudo code of EMCTS with EUCT, where we suppress dependence on the model $\hat{M}$ for notation simplicity. Extensions to MCTS are marked in blue. In AZ, with a true reward model, EMCTS will use $\mathbb{V}[\hat{R}(s, a)] = 0, \forall (s, a) \in \mathcal{S} \times \mathcal{A}$, but still use $\mathbb{V}[\hat{V}(s)] \geq 0$.

To conclude this section we note that while in this work we motivate and later evaluate EMCTS from the perspective of deep exploration, EMCTS is not limited to this use case. Our method introduces to MCTS-based algorithms such as A/MZ a novel capability to estimate the epistemic uncertainty in the value predictions during and post search, which may be attractive for different purposes. Noted example are: (i) reducing over estimation errors with lower-bound value targets of the form $q_{\hat{M}}(s, a) - \beta\sqrt{\mathbb{V}[q_{\hat{M}}(s, a)]}$. This approach is popular in model free AC methods implicitly (Fujimoto et al., 2018; Haarnoja et al., 2018)

Algorithm 1 EMCTS with EUCT. Requires f, r, v^π and uncertainty estimators $\mathbb{V}[\hat{V}], \mathbb{V}[\hat{R}]$

function EMCTS(state s, β) ▷ $\beta = 0$ for unmodified MCTS exploitation episodes
  while within computation budget **do**
    SELECT(s, β) ▷ traverses tree from root $s_0 := s$ and adds new leaf
  return action a drawn from $\pi(a_0|s) = \frac{N(s_0,a)}{\sum_{a'} N(s_0,a')}$ ▷ MCTS action selection

function SELECT(node s_k, β)
  $a_k \leftarrow \arg\max_{a \in A} q^\beta(s_k, a) + C_{\text{UCT}} \sqrt{\frac{2 \log(\sum_{a'} N(s_k,a'))}{N(s_k,a)}}$ ▷ Equation 9
  if a_k already expanded **then** SELECT($f(s_k, a_k)$, β) ▷ traverses tree
  else EXPAND(s_k, a_k) ▷ adds new leaf

function EXPAND(node s_k, not yet expanded action a_k)
  $s_{k+1}, \hat{V}(s_{k+1}), \hat{R}(s_k, a_k) \leftarrow$ Execute MCTS expansion ▷ creates a new leaf s_{k+1}
  Estimate epistemic variance of reward $\mathbb{V}[\hat{R}(s_k, a_k)]$ and store it in node s_{k+1}.
  Estimate epistemic variance of value $\mathbb{V}[\hat{V}(s_{k+1})]$ and store it in node s_{k+1}.
  BACKUP($s_{k+1}, \hat{V}(s_{k+1}), \mathbb{V}[\hat{V}(s_{k+1})]$) ▷ updates the tree values & value variances

function BACKUP(node s_{k+1}, ret. $\nu(s_{k+1}, a_{k+1})$, ret. unc. $\mathbb{V}[\nu(s_{k+1}, a_{k+1})]$)
  $s_k, a_k, \nu(s_k, a_k) \leftarrow$ Execute MCTS backup step ▷ updates $q(s_k, a_k)$, $N(s_k, a_k)$
  $\mathbb{V}[\nu(s_k, a_k)] \leftarrow \mathbb{V}[\hat{R}(s_k, a_k)] + \gamma^2 \mathbb{V}[\nu(s_{k+1}, a_{k+1})]$ ▷ Equation 12
  $\sigma_q(s_k, a_k) \leftarrow \sigma_q(s_k, a_k) + \frac{\sqrt{\mathbb{V}[\nu(s_k,a_k)]} - \sigma_q(s_k,a_k)}{N(s_k,a_k)}$ ▷ Equation 13
  if $k > 0$ **then** BACKUP($s_k, \nu(s_k, a_k), \mathbb{V}[\nu(s_k, a_k)]$)

and explicitly (Ciosek et al., 2019), as well as in model-based RL (Zhou et al., 2020) and is generally very common in offline RL (Kumar et al., 2020; Ghasemipour et al., 2022). This makes EMCTS an espeically attractive candidate to enhance A/MZ's Reanalyze for offline-RL or off-policy target generation (Schrittwieser et al., 2021). We propose to track the objective $q_{\hat{M}}(s, a) - \beta \sqrt{\mathbb{V}[q_{\hat{M}}(s, a)]}$ in search, i.e. to use EP/UCT but with $\beta < 0$. (ii) Weighting value and policy losses by the estimate of the epistemic uncertainty in the value which was successful in online as well as offline RL (Lee et al., 2021; Wu et al., 2021). The uncertainty of the value of the root of EMCTS can be used for this purpose.

3.4 Search with a Learned Transition Model

Learned transition models introduce several challenges from the perspective of EMCTS: (i) estimating epistemic uncertainty in the possibly-abstracted planning space, (ii) propagating the uncertainty forward during search as future transitions become less certain, in addition to backwards, and (iii) propagating it in such a way that maintains the UCB constructed in Theorem 1. Multiple methods to overcome (i) have been successful in previous works, see Henaff (2019); Sekar et al. (2020). In Appendix A.3 we propose an approach to overcome (ii) and (iii) as well as discuss the challenges in more detail. In practice however, in Section 5 we include experiments where a MZ agent fitted with a reliable uncertainty estimator for $\mathbb{V}[\hat{R}(s, a)]$ successfully demonstrates deep exploration without accounting for (ii) and (iii).

4 Related Work

Search for exploration was used successfully in a number of previous works, see Jaksch et al. (2010), Sun et al. (2011), Hester & Stone (2012), Shyam et al. (2019), Henaff (2019), Sekar et al. (2020), Lambert et al. (2022) and Luis et al. (2023). We add to this line of work EMCTS: designed for MCTS (and planning trees in general), practical and based in theory. Tesauro et al. (2010) develop a Bayesian approach for aleatoric uncertainty propagation in MCTS. POMCP (Silver & Veness, 2010), POMCPOW (Sunberg & Kochenderfer, 2018) and BOMCP (Mern et al., 2021) extend MCTS to POMDPs with a probabilistic Bayesian belief state at the nodes using a probabilistic model, while Stochastic MuZero (Antonoglou et al., 2022) extends MuZero to the stochastic transitions setting by replacing the deterministic transition function with a Vector Quantised Variational AutoEncoder (van den Oord et al.,

2017). In these works, epistemic uncertainty is not distinguished or used for exploration. Latent disagreement ensembles (Lakshminarayanan et al., 2017; Ramesh et al., 2022) offer a popular alternative to RND and counts. Wasserstein Temporal Difference (WTD, Metelli et al., 2019) offers an alternative to UBE (O'Donoghue et al., 2018) for estimating value uncertainty, using Wasserstein Barycenters (Agueh & Carlier, 2011) to update a posterior over Q functions in place of a standard Bayesian update. While UBE was criticized by Janz et al. (2019) for being insufficient for deep exploration with posterior-sampling based RL (PSRL, Osband et al., 2013), we note that the same shortcomings do not apply when UBE is used for UCB-based exploration.

5 Experiments

The task of writing code and finding new algorithms, where AZ has recently made world real-world impact (Fawzi et al., 2022; Mankowitz et al., 2023), is natural to formulate using sparse-reward environments where the actions of the agent are operations and reward is received when a correct program is completed. Such environments represent a hard exploration challenge: the state space is often exponential $|\mathcal{A}|^L$ in the actions $\mathcal{A}$ and the maximum length of the program L, and while the number of possible solutions is generally unknown, it is in most cases very small compared to the number of possible sequences of operations. To evaluate the contribution of EMCTS in such real-world tasks we conduct experiments in the one-instruction Assembly programming language SUBLEQ (Section 5.1).

To verify that EMCTS demonstrates deep exploration and benefits from search, we conduct experiments in the commonly used hard-exploration benchmark Deep Sea (Osband et al., 2020) (Section 5.2). Specifically, we are interested in the following:
RQ I *Does EMCTS paired with an epistemic search policy demonstrate deep exploration with both AZ as well as MZ and in the presence of stochastic as well as deterministic rewards?*
RQ II *Is there benefit in search with EMCTS for deep exploration, compared to an otherwise equivalent approach that does not use search, and if so, is the benefit retained in the presence of the learned transition model of MZ?*
We also conduct an ablation study on the exploration parameter β to verify that the agent can learn stably from the possibly very-off-policy exploratory data provided by deep exploration.

5.1 SUBLEQ Experiments

SUBLEQ is a Turing-complete (excepting for finite memory) one-instruction programming language (Mazonka & Kolodin, 2011). Because there is only one instruction, writing code in SUBLEQ summarizes to writing a sequence of memory addresses. We model the action space with $\mathcal{A} = \{0, \dots, N-1\}$, for N the size of the memory. The agent is rewarded with 1.0 when the sequence of addresses specifies a program that solves the task, evaluated on a set of test cases. The reward is otherwise zero. The observation space constitutes of an example input and corresponding desired output, the program written so far, and the state of the input and output after executing the program on the example input. We limit the memory size to 16 and the maximum program length to 10 resulting in a state space of size $\leq 16^{10} \approx 1$ *trillion* unique states. We present results on two tasks: An easier task of outputting the negated input for positive inputs (Negate Positives), and a harder task of implementing the Identity Function. An implementation of Negate Positives in SUBLEQ of length 2 is known, which means finding a solution without prior knowledge requires searching on the order of 16^2 unique states. To the Identity Function an implementation of length 6 is known which suggests searching on the order of $16^6 \approx 16$ million unique states. We provide a more detailed introduction to SUBLEQ, a description of the environment and the tasks in Appendix C. The results are presented in Figure 1.

We compare two variations of EMCTS with AZ (**E-AZ**) to the **AZ** baseline. Both variations of E-AZ are able to solve the harder task in a much smaller number of samples than AZ. To estimate $\mathbb{V}[\hat{R}(\cdot, \cdot, s')]$ one variation of E-AZ uses a hash based visit count that takes as input the complete state s', allowing the agent to, in principle, avoid searching the same state twice. The other E-AZ variation hashes only part of the state: the example input-output, before and after execution (IO hash), directing the agent to search actions that have an

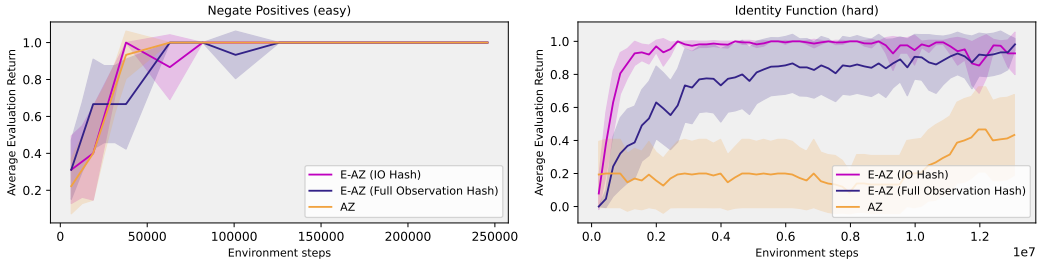

Figure 1: Sample efficiency. Left: the easy SUBLEQ *Negate Positives* task. Right: the harder *Identity Function* task. Mean of 15 seeds, two standard errors.

effect on the output of the program. As expected from a UCB-based method, E-AZ benefits from the more-appropriate uncertainty estimator and finds a correct program much earlier with the IO hash.

All agents use Danihelka et al. (2022)'s approach that combines Gumbel noise with Sequential Halving at the root and a modern variation of PUCT at non-root nodes. For E-AZ variations, in online search (when selecting actions in the environment) the tree search policies use $q^{\beta}_{\hat{M}}$ (11) in place of $q_{\hat{M}}$ and an exploration-prior-policy π_e (See Section 3.2). To select actions during evaluation, as well as when generating targets with Reanalyze, the search uses EMCTS to propagate the uncertainty, but uses the exploitation prior policy π and $q_{\hat{M}}$ in the search objective, in the standard manner of MCTS search. The policy $\pi_e(s)$ is trained with cross entropy loss to fit the softmax across actions over $q^{\beta}(s, \cdot)$ at the root.

5.2 Deep Sea Experiments

The Deep Sea environment is structured as a grid, where at each time step the agent chooses between two actions, goes right or left and one row down. There is a reward $r_{goal} = 1.0$ at the bottom right corner, the agent starts at the top left corner and thus the probability of randomly finding the unique optimal trajectory decays exponentially with the size of the grid. Every transition along the optimal trajectory receives a negative reward that is negligible in comparison to the goal reward, but is otherwise the only reward the agent sees, actively discouraging exploration along the optimal path. The action mappings are randomized such that the effect of the same action is different in different states, preventing the agent from generalizing across actions. To evaluate EMCTS in the presence of deterministic transitions and stochastic rewards that more generally align with the assumptions in Section 3, we include a custom stochastic-reward variation of Deep Sea where the goal reward $r_{goal} \sim \mathcal{N}(1, 1)$, and a reward $r_{mislead} \sim \mathcal{N}(0, 1)$ is given at the bottom left corner. The agent must explore both transitions a sufficient number of times to correctly identify the larger mean reward. Results are presented in Figure 2.

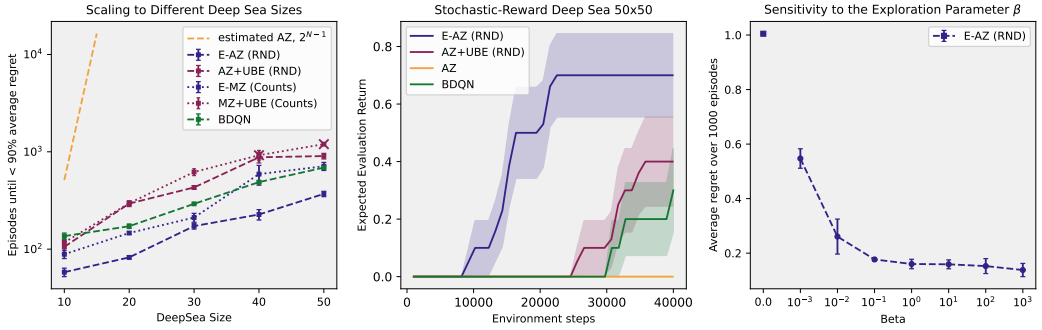

Figure 2: Left: Scaling to growing Deep Sea sizes, 5 seeds per point. Only 2 seeds of MZ+UBE were able to solve size 40, and none size 50 within the training budget, both marked with an X. Middle: Stochastic-reward Deep Sea 50x50, 10 seeds. Right: The effect of the exploration parameter in Deep Sea 30x30, 3 seeds per point. Mean and standard error.

The left subplot shows sample complexity measured as number of interactions until 90% regret is reached against the size of the Deep Sea environment to demonstrate that the scaling is sub-exponential as expected from deep-exploration methods. The middle subplot presents expected return in evaluation episodes in the stochastic-reward variation of Deep Sea. In the right subplot, the sensitivity to the exploration parameter β is studied and the ability of the agent to learn stably from off-policy data. In Appendix B we include additional results, including a visualization of the uncertainty estimated by EMCTS.

To answer the research questions the following agents based in A/MZ are compared: (i) **E-A/MZ** (purple, dashed for AZ and dotted for MZ), our method. The agents use EMCTS with counts/RND to estimate reward and transition uncertainty and UBE to estimate value uncertainty, search with EUCT and act with the action with the most visitations at the root. (ii) Baseline **AZ** (orange) explores by sampling actions proportionally to visitations at the MCTS root. AZ is included for reference, to demonstrate that indeed Deep Sea cannot be solved in reasonable time with random exploration, and is not able to solve any of the environments in the alloted training steps, with the exception of one seed in the smallest environment size, and that only because the initial replay buffer by chance already contained a trajectory that reached the goal. We include the exponentially-scaling performance expected of random-exploration based methods such as AZ, for reference. (iii) Last, **A/MZ+UBE** (red) acts with a similar UCB-exploration objective to that of EP/UCT, but does not search with the uncertainty (see Appendix D.8). AZ-based agents are given access to the true transition model. The value model is learned, as well as the reward model to investigate the behavior under the most general setting considered in Section 3.3. All A/MZ agents are trained with targets generated by Reanalyze. For reference performance on Deep Sea, we include Bootstrapped DQN (**BDQN,** Osband et al., 2018), a popular model-free, non-search based deep exploration approach that relies on an ensemble to drive deep exploration directly. For full implementation details see Appendix D.

As expected for deep exploration methods, agents that are informed with respect to epistemic uncertainty (E-A/MZ, A/MZ+UBE and BDQN) demonstrate sample efficiency that scales sub-exponentially with environment size (Figure 2, left). In addition, E-AZ is able to solve Deep Sea in the presence of stochastic rewards (Figure 2, middle). This answers **RQ I**: EMCTS successfully demonstrates deep exploration in the presence of learned value, reward and transition models as well as both deterministic and stochastic rewards. E-A/MZ (Figure 2, left and middle plots, purple) demonstrate significant improvement in sample efficiency over the equivalent agents that do not use search (red), in deterministic and stochastic reward environments and even with the learned transition model of MZ (dotted line). This answers **RQ II**: EMCTS demonstrates benefits from search for uncertainty estimation and exploration which is retained in the presence of MZ's learned transition dynamics' model. The low regret in evaluation with exponentially increasing values of β (Figure 2, right) demonstrates that the agent can stably learn the optimal policy even in the presence of off-policy exploratory data.

6 Conclusions

In this work we present EMCTS, a novel, practical and theoretically motivated method to incorporate the epistemic uncertainty from learned models into MCTS, as well as harness the search for deep exploration. AZ paired with EMCTS (E-AZ) achieves significantly higher sample efficiency in the sparse-reward, challenging task of programming in the Assembly language SUBLEQ, compared to baseline AZ. In the popular hard-exploration benchmark Deep Sea, E-A/MZ demonstrate deep exploration by solving variations of the task which cannot be solved by baseline A/MZ at all in a reasonable amount of samples. EMCTS' search demonstrates significantly improved epistemic uncertainty estimation through more sample efficient exploration over an otherwise equivalent method that does not use search. With EMCTS, A/MZ are much better equipped for sparse-reward and hard-exploration environments, which come up in realistic settings such as algorithm design, where AZ has already made significant advances. By making A/MZ uncertainty aware, EMCTS is also promising for settings that require improved reliability in the face of the unknown such as offline RL and off-policy target generation.

Acknowledgements

We would like to thank Marco Loog, Frank van der Meulen, Itamar Sher, Moritz Zanger, Pascal van der Vaart, Joery de Vries & Jinke He 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), and partially funded by the Dutch Research Council (NWO) project *Reliable Out-of-Distribution Generalization in Deep Reinforcement Learning* with project number OCENW.M.21.234.

References

Martial Agueh and Guillaume Carlier. Barycenters in the wasserstein space. *SIAM J. Math. Anal.*, 43(2):904–924, 2011. doi: 10.1137/100805741.

Ioannis Antonoglou, Julian Schrittwieser, Sherjil Ozair, Thomas K. Hubert, and David Silver. Planning in stochastic environments with a learned model. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 263–272. PMLR, 2017.

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

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Cameron Browne, Edward Jack Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez Liebana, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Trans. Comput. Intell. AI Games*, 4(1):1–43, 2012. doi: 10.1109/TCIAIG.2012.2186810.

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.

Kamil Ciosek, Quan Vuong, Robert Tyler Loftin, and Katja Hofmann. Better exploration with Optimistic Actor Critic. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 1785–1796, 2019.

Ivo Danihelka, Arthur Guez, Julian Schrittwieser, and David Silver. Policy improvement by planning with gumbel. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent

Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020. URL `http://github.com/deepmind`.

Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis, and Pushmeet Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022. doi: 10.1038/s41586-022-05172-4.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1582–1591. PMLR, 2018.

Seyed Kamyar Seyed Ghasemipour, Shixiang Shane Gu, and Ofir Nachum. Why so pessimistic? estimating uncertainties for offline RL through ensembles, and why their independence matters. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Mikael Henaff. Explicit explore-exploit algorithms in continuous state spaces. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 9372–9382, 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-EPIROB 2012, San Diego, CA, USA, November 7-9, 2012*, pp. 1–6. IEEE, 2012. doi: 10.1109/DEVLRN.2012.6400802.

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.

Ola Hössjer and Arvid Sjölander. Sharp lower and upper bounds for the covariance of bounded random variables. *Statistics & Probability Letters*, 182:109323, 2022. ISSN 0167-7152. doi: https://doi.org/10.1016/j.spl.2021.109323.

Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *J. Mach. Learn. Res.*, 11:1563–1600, 2010. doi: 10.5555/1756006.1859902.

David Janz, Jiri Hron, Przemyslaw Mazur, Katja Hofmann, José Miguel Hernández-Lobato, and Sebastian Tschiatschek. Successor uncertainties: Exploration and uncertainty in temporal difference learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 4509–4518, 2019.

Chi Jin, Zeyuan Allen-Zhu, Sébastien Bubeck, and Michael I. Jordan. Is Q-learning provably efficient? In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò

Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 4868–4878, 2018.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I. Jordan. Provably efficient reinforcement learning with linear function approximation. *Math. Oper. Res.*, 48(3):1496–1521, 2023. doi: 10.1287/MOOR.2022.1309.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou (eds.), *Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006, Proceedings*, volume 4212 of *Lecture Notes in Computer Science*, pp. 282–293. Springer, 2006. doi: 10.1007/11871842_29.

Sotetsu Koyamada, Shinri Okano, Soichiro Nishimori, Yu Murata, Keigo Habara, Haruka Kita, and Shin Ishii. Pgx: Hardware-accelerated parallel game simulators for reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6402–6413, 2017.

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.

Tor Lattimore and Csaba Szepesvari. *Bandit Algorithms*. Cambridge University Press, 2017.

Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. SUNRISE: A simple unified framework for ensemble learning in deep reinforcement learning. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6131–6141. PMLR, 2021.

Owen Lockwood and Mei Si. A review of uncertainty for deep reinforcement learning. In Stephen G. Ware and Markus Eger (eds.), *Proceedings of the Eighteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2022, Pomona, CA, USA, October 24-28, 2022*, pp. 155–162. AAAI Press, 2022.

Carlos E. Luis, Alessandro G. Bottero, Julia Vinogradska, Felix Berkenkamp, and Jan Peters. Model-based uncertainty in value functions. In Francisco J. R. Ruiz, Jennifer G. Dy, and Jan-Willem van de Meent (eds.), *International Conference on Artificial Intelligence and Statistics, 25-27 April 2023, Palau de Congressos, Valencia, Spain*, volume 206 of *Proceedings of Machine Learning Research*, pp. 8029–8052. PMLR, 2023.

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.

Daniel J. Mankowitz, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, Shariq Iqbal, Jean-Baptiste Lespiau, Alex Ahern, Thomas Koppe, Kevin Millikin, Stephen Gaffney, Sophie Elster, Jackson Broshear, Chris Gamble, Kieran Milan, Robert Tung, Minjae Hwang, Taylan Cemgil, Mohammadamin Barekatain, Yujia Li, Amol Mandhane, Thomas Hubert, Julian Schrittwieser, Demis Hassabis, Pushmeet Kohli, Martin Riedmiller, Oriol Vinyals, and David Silver. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618(7964):257–263, 2023. doi: 10.1038/s41586-023-06004-9.

Oleg Mazonka and Alex Kolodin. A simple multi-processor computer based on subleq. *arXiv preprint arXiv:1106.2593*, 2011.

John Mern, Anil Yildiz, Zachary Sunberg, Tapan Mukerji, and Mykel J. Kochenderfer. Bayesian optimized Monte-Carlo planning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 11880–11887. AAAI Press, 2021. doi: 10.1609/AAAI.V35I13.17411.

Alberto Maria Metelli, Amarildo Likmeta, and Marcello Restelli. Propagating uncertainty in reinforcement learning via Wasserstein barycenters. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 4335–4347, 2019.

Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. Model-based reinforcement learning: A survey. *Found. Trends Mach. Learn.*, 16(1):1–118, 2023. doi: 10.1561/2200000086.

Gergely Neu and Ciara Pike-Burke. A unifying view of optimism in episodic reinforcement learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Brendan O'Donoghue, Ian Osband, Rémi Munos, and Volodymyr Mnih. The uncertainty Bellman equation and exploration. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3836–3845. PMLR, 2018.

Ian Osband, Daniel Russo, and Benjamin Van Roy. (More) efficient reinforcement learning via posterior sampling. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pp. 3003–3011, 2013.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 8626–8638, 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 S. Sutton, David Silver, and Hado van Hasselt. Behaviour suite for reinforcement learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Tiberiu Popoviciu. Sur les équations algébriques ayant toutes leurs racines réelles. *Mathematica*, 9(129-145):20, 1935.

Aditya A. Ramesh, Louis Kirsch, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Exploring through random curiosity with general value functions. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 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.

Julian Schrittwieser, Thomas Hubert, Amol Mandhane, Mohammadamin Barekatain, Ioannis Antonoglou, and David Silver. Online and offline reinforcement learning by planning with a learned model. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 27580–27591, 2021.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8583–8592. PMLR, 2020.

Pranav Shyam, Wojciech Jaskowski, and Faustino Gomez. Model-based active exploration. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5779–5788. PMLR, 2019.

David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. In John D. Lafferty, Christopher K. I. Williams, John Shawe-Taylor, Richard S. Zemel, and Aron Culotta (eds.), *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*, pp. 2164–2172. Curran Associates, Inc., 2010.

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.

Malcolm J. A. Strens. A Bayesian framework for reinforcement learning. In Pat Langley (ed.), *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pp. 943–950. Morgan Kaufmann, 2000.

Yi Sun, Faustino J. Gomez, and Jürgen Schmidhuber. Planning to be surprised: Optimal Bayesian exploration in dynamic environments. In Jürgen Schmidhuber, Kristinn R. Thórisson, and Moshe Looks (eds.), *Artificial General Intelligence - 4th International Conference, AGI 2011, Mountain View, CA, USA, August 3-6, 2011. Proceedings*, volume

6830 of *Lecture Notes in Computer Science*, pp. 41–51. Springer, 2011. doi: 10.1007/978-3-642-22887-2_5.

Zachary N. Sunberg and Mykel J. Kochenderfer. Online algorithms for POMDPs with continuous state, action, and observation spaces. In Mathijs de Weerdt, Sven Koenig, Gabriele Röger, and Matthijs T. J. Spaan (eds.), *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling, ICAPS 2018, Delft, The Netherlands, June 24-29, 2018*, pp. 259–263. AAAI Press, 2018.

Maciej Swiechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mandziuk. Monte Carlo Tree Search: a review of recent modifications and applications. *Artif. Intell. Rev.*, 56(3): 2497–2562, 2023. doi: 10.1007/S10462-022-10228-Y.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. #exploration: A study of count-based exploration for deep reinforcement learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 2753–2762, 2017.

Gerald Tesauro, V. T. Rajan, and Richard B. Segal. Bayesian inference in Monte-Carlo Tree Search. In Peter Grünwald and Peter Spirtes (eds.), *UAI 2010, Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, July 8-11, 2010*, pp. 580–588. AUAI Press, 2010.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6306–6315, 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 Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11319–11328. PMLR, 2021.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering Atari games with limited data. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 25476–25488, 2021.

Qi Zhou, Houqiang Li, and Jie Wang. Deep model-based reinforcement learning via estimated uncertainty and conservative policy optimization. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 6941–6948. AAAI Press, 2020. doi: 10.1609/AAAI.V34I04.6177.

A Additional Results

A.1 Proof of Theorem 1

For clarity, let us again define π^* as the policy that is optimal in the true environment $\pi^* = \arg\max_\pi Q^\pi_m$, and $\pi^*_{\hat{M}}$ the policy that is optimal in a specific model $\hat{M}$, that is $\pi^*_{\hat{M}} = \arg\max_\pi Q^\pi_{\hat{M}}$. By the definition of the model, we have $\mathbb{E}[\hat{M}] = m$, where m is the true model of the environment. Therefore $Q^* =: Q^{\pi^*}_m = Q^{\pi^*}_{\mathbb{E}[\hat{M}]} = \mathbb{E}_{\hat{M}}[Q^{\pi^*}_{\hat{M}}]$, by linearity of the value in the reward function. By Chebyshev's inequality:

$$P\Big(\mathbb{E}_{\hat{M}}[Q^{\pi^*}_{\hat{M}}] \leq Q^{\pi^*}_{\hat{M}} + \frac{1}{\sqrt{\delta}}\sqrt{\mathbb{V}[Q^{\pi^*}_{\hat{M}}]}\Big) \geq 1 - \delta. \tag{14}$$

The right-hand side term $Q^{\pi^*}_{\hat{M}} + \frac{1}{\sqrt{\delta}}\sqrt{\mathbb{V}[Q^{\pi^*}_{\hat{M}}]}$ is hard to estimate among other reasons because the optimal policy is not known. But it can be upper bounded as follows:

$$Q^{\pi^*}_{\hat{M}} + \frac{1}{\sqrt{\delta}}\sqrt{\mathbb{V}[Q^{\pi^*}_{\hat{M}}]} \leq \max_\pi Q^\pi_{\hat{M}} + \frac{1}{\sqrt{\delta}}\sqrt{\mathbb{V}[Q^\pi_{\hat{M}}]},$$

which holds because the right-hand side is the maximum possible policy in a set that contains π^*.

Putting the entire set of inequalities together, we arrive at:

$$P\Big(Q^* = \mathbb{E}_{\hat{M}}[Q^{\pi^*}_{\hat{M}}] \leq Q^{\pi^*}_{\hat{M}} + \tfrac{1}{\sqrt{\delta}}\sqrt{\mathbb{V}[Q^{\pi^*}_{\hat{M}}]} \leq \max_\pi Q^\pi_{\hat{M}} + \tfrac{1}{\sqrt{\delta}}\sqrt{\mathbb{V}[Q^\pi_{\hat{M}}]}\Big) \geq 1 - \delta, \tag{15}$$

$$\text{and thus:} \quad P\Big(Q^* \leq \max_\pi Q^\pi_{\hat{M}} + \tfrac{1}{\sqrt{\delta}}\sqrt{\mathbb{V}[Q^\pi_{\hat{M}}]}\Big) \geq 1 - \delta, \tag{16}$$

QED Theorem 1.

A.2 Derivation of the Upper Bound on $\mathbb{V}[q_{\hat{M}}(s_k, a)]$

By definition,

$$\mathbb{V}[q_{\hat{M}}(s_k, a)] = \tfrac{1}{n^2}\mathbb{V}\Big[\sum_{i=1}^n \nu^i(s_k, a)\Big] = \tfrac{1}{n^2}\sum_{i,j=1}^n \mathrm{Cov}\big(\nu^i(s_k, a), \nu^j(s_k, a)\big)\,. \tag{17}$$

In A/MZ it is standard to learn one set of deterministic models $\hat{V}, \hat{R}$ and use them throughout planning, and therefore ν^i, ν^j cannot be assumed to be independent $\forall i \neq j$. Had they been independent, one would compute the variance of the sum directly with the sum of the variances. Instead, we use the inequality for covariances with known variances (Hössjer & Sjölander, 2022): $\mathrm{Cov}[X, Y] \leq \sqrt{\mathbb{V}[X]\mathbb{V}[Y]}$ to upper bound the variance $\mathbb{V}[q_{\hat{M}}(s_k, a)]$:

$$\mathbb{V}[q_{\hat{M}}(s_k, a)] = \tfrac{1}{n^2}\mathbb{V}\Big[\sum_{i=1}^n \nu^i(s_k, a)\Big] = \tfrac{1}{n^2}\sum_{i,j=1}^n \mathrm{Cov}\big(\nu^i(s_k, a), \nu^j(s_k, a)\big) \tag{18}$$

$$\leq \tfrac{1}{n^2}\sum_{i,j=1}^n \sqrt{\mathbb{V}[\nu^i(s_k, a)]\,\mathbb{V}[\nu^j(s_k, a)]} \;=\; \Big(\tfrac{1}{n}\sum_{i=1}^n \sqrt{\mathbb{V}[\nu^i(s_k, a)]}\Big)^2. \tag{19}$$

In other words, the standard deviation of the averaged backup $\sqrt{\mathbb{V}[q_{\hat{M}}(s_k, a)]}$ is upper bounded with the averaged standard deviation across backups $\frac{1}{n}\sum_{i=1}^n \sqrt{\mathbb{V}[\nu^i(s_k, a)]}$. The first inequality is due to the inequality for covariances with known variances, and the following equality is a sum of squares.

A.3 EMCTS with learned transition dynamics

When the transition function $\hat{f}$ is learned, we must formulate it as a random variable $\hat{F}$ as well. We note that in this case $S' \sim \hat{F}(S, a)$ is a random variable that depends on the distribution of the previous state S propagated through the uncertain transition function $\hat{F}$, and similarly the distribution of S' will propagate through the reward and value and future transition predictions $\hat{R}(S', a), \hat{V}^\pi_{\hat{M}}(S'), \hat{F}(S', a)$. In other words, in order to estimate $\mathbb{V}[Q^\pi_{\hat{M}}]$

we need to account for propagation of the variance through the Markov chain, which is an expensive and non-trivial process. In addition, as the value is non-linear in the transition dynamics f, Theorem 1 does not apply and Equation 8 is not an upper bound with a specific probability. Finally, to extend EMCTS to MZ fully, we must account for the fact that the dynamics model learned by MZ is not operating in the true state space of the environment, but in a value-equivalent abstraction. To extend EMCTS to the learned dynamics case, we will assume that it is possible to estimate epistemic uncertainty in the abstracted state space $\hat{\mathcal{S}}$ in a meaningful way. This problem can be circumvented by driving additional losses through the learned model that incentivize distinguishing between unique states in latent space (Henaff, 2019), or by learning an auxiliary dynamics model to distinguish between novel and observed starting-states-and-action-sequences, which has been used successfully by Sekar et al. (2020). We will therefore use the notation of states s or S in the tree, whether they are in the true state space of the environment $\mathcal{S}$ or the abstracted space of MZ $\hat{\mathcal{S}}$. In addition, since standard MZ plans with a deterministic transition model, we extend EMCTS to the setting where either the underlying transition dynamics f are deterministic, or the abstracted deterministic transition function is sufficiently meaningful. For simplicity, let us further assume that the state space $\mathcal{S}$ is continuous, and the starting state distribution ρ is over a finite domain.

To circumvent the problem of the propagation of state uncertainty through the Markov chain, we propose a cheap and simple maximally-optimistic alternative upper-bound approximation for $\mathbb{V}[Q^{\pi}_{\hat{M}}]$. We note that: (i) $\hat{F}, \hat{R}$ are trained on the same data and thus when the epistemic variance over one of the two is maximal the other can be expected to be maximal as well. (ii) We have assumed a deterministic true transition function f and thus the variance in $\hat{F}$ can be modelled as maximal on the unknown and zero on the known. (iii) If the uncertainty in the prediction of any state $\mathbb{V}[\hat{F}(S_k, a)] \geq 0$, we can expect the uncertainty in all future predictions S_j along this trajectory $j > k$ to be associated with maximal uncertainty. We use these observations to formulate the following simple approximation:

$$\mathbb{V}[\hat{R}(S_k, a)] \approx \mathbb{V}[\hat{R}(s_k, a)],\ \mathbb{V}[\hat{V}^{\pi}_{\hat{M}}(S_k)] \approx \mathbb{V}[\hat{V}^{\pi}_{\hat{M}}(s_k)],\ \forall (s, a) \in \mathcal{D} \tag{20}$$

$$\mathbb{V}[\hat{R}(S_j, a)] \approx r^2_{max},\ \mathbb{V}[\hat{V}^{\pi}_{\hat{M}}(S_j)] \approx (\tfrac{1}{1-\gamma} r_{max})^2,\ \text{where } j \geq k \text{ and } \forall (s, a) \notin \mathcal{D} \tag{21}$$

That is, we propose to ignore any uncertainty in transition along zero-uncertainty trajectories, and with the first uncertain transition, to assume all uncertainties are maximal for all predictions in the rest of the trajectory. To identify whether $(s, a) \notin \mathcal{D}$ we can directly use $\mathbb{V}[\hat{R}(s, a)]$, where $\mathbb{V}[\hat{R}(s, a)] \approx r^2_{max}$ indicates $(s, a) \notin \mathcal{D}$. This will guarantee that (given that the mechanism to identify unobserved transitions is sufficiently reliable) the agent remains sufficiently optimistic and the planning objective of EP/UCT remains an upper bound on Q^*. We note that in practice, while this approach is theoretically sound, the results of the experiments in Section 5.2, in the presence of reliable transition-uncertainty, motivate that it is not necessary and the regular setup of EMCTS is sufficient for deep exploration even in the presence of a learned, value-equivalent-abstraction-based transition model, given that $\hat{R}(s, a)$ is a sufficiently reliable estimator of novelty in the environment.

B Additional Experiments

We include an example comparison of the uncertainty estimated by EMCTS to that of the uncertainty estimator used by EMCTS, the UBE network head, in Figure 3. The E-AZ agent used in this experiment uses the true transition model but a learned reward as well as value functions, matching the setup of Section 3.1. The Deep Sea environment is presented as a grid, where states are the cells including and below the diagonal. Bold blue in the grids in the bottom row (inverse counts) indicates unvisited states. By averaging across multiple predictions in search, the uncertainty estimated by EMCTS (top row) is much more varied, more accurately associating less or more uncertainty with states that lead into observed / unobserved trajectories respectively, compared to the UBE predictions (middle row). Most importantly from the perspective of exploration, EMCTS associates larger uncertainty with states that lead into unobserved directions much more consistently than the single predictions of UBE for each state (for an easily visible example, t= 2000, top of the diagonal).

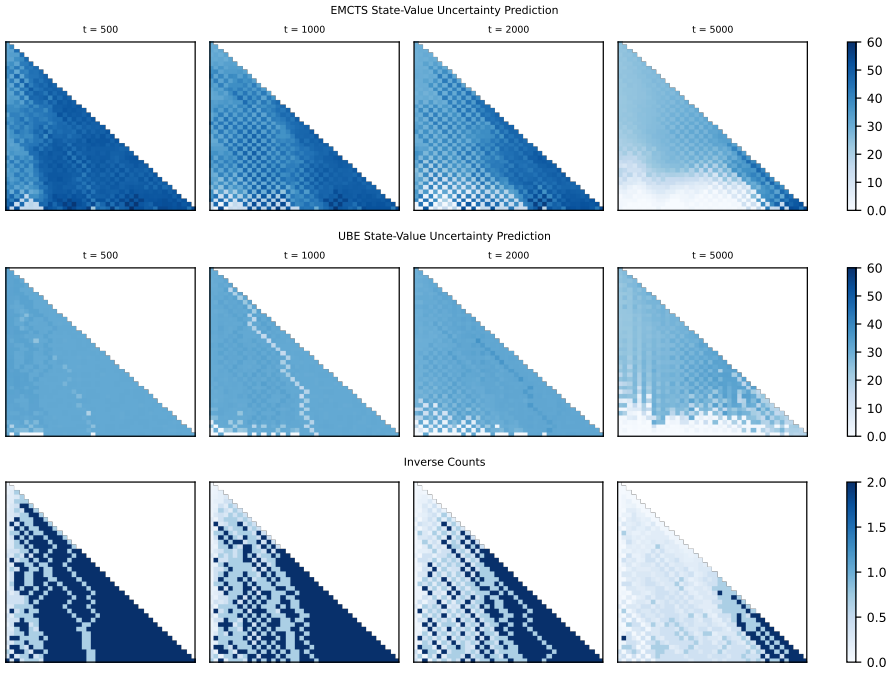

Figure 3: Heat maps over states in DeepSea 40 by 40 at different times (columns) during an example training run of EMCTS with an AZ transition model. Upper row: value uncertainty at the EMCTS root node. Middle row: single prediction of UBE at each state. Lower row: inverse visitation counts as reliable local uncertainty, where score of 2.0 represents unvisited.

We include an addition detailed evaluation curves for E-A/MZ, A/MZ and A/MZ+UBE in Figure 4, as well as rate-of-exploration (number of unique states encountered per interaction with the environment). An additional variation of MZ is included in this Figure, where instead of value-equivalent abstraction the model is trained with a reconstruction loss to match the true transition function and observation space of the environment, such that $\hat{f}(s,a) = \hat{s}' \approx s'$. By training the RND estimator only on true transitions (s,a) and evaluating it online on transitions predicted by the learned transition model $(\hat{s}', a')$, $\hat{s}' = \hat{f}(\hat{s}, a)$, this agent is incentivized to estimate the uncertainty over every uncertain transition $(s,a) \notin \mathcal{D}$ as maximal, implicitly implementing the approach described in Appendix A.3. For implementation details see Appendix D.3. In all figures, E-A/MZ outperforms the other agents both in rate of exploration as well as in their ability to learn to reach the goal. The most interesting behavior is perhaps that of the reconstruction based model, that does not search the environment significantly faster than the other baselines, and yet learns to reach the goal much earlier (bottom row in Figure 4). We hypothesize that due to the search, the uncertainty estimated by the agent is more reliable, resulting in identifying the correct action that leads into novel states, *more times in a row*. Just visible on the right-hand plot, one can see that indeed the purple curve remains the highest for quite a while, before all others curves match it, despite all curves being very close throughout most of the training.

We include a table evaluating interactions-to-goal on Deep Sea 40x40 for all agents. The results demonstrate that even when the learned dynamics model is not designed for planning (anchored model, third block, Table 1), EMCTS is able to find the goal much faster.

Finally, we include a comparison between E-AZ and AZ on MinAtar in Figure 5. The uncertainty estimator is the full-Hash used for SUBLEQ, which uses hash based counting to distinguish between any two unique states. This demonstrates that even when the uncertainty estimator is unsuited, and even without tuning the exploration parameter $\beta = 0$ (in Figure 5 $\beta = 1.0$ as in the SUBLEQ experiments), EMCTS can compare well to the baseline.

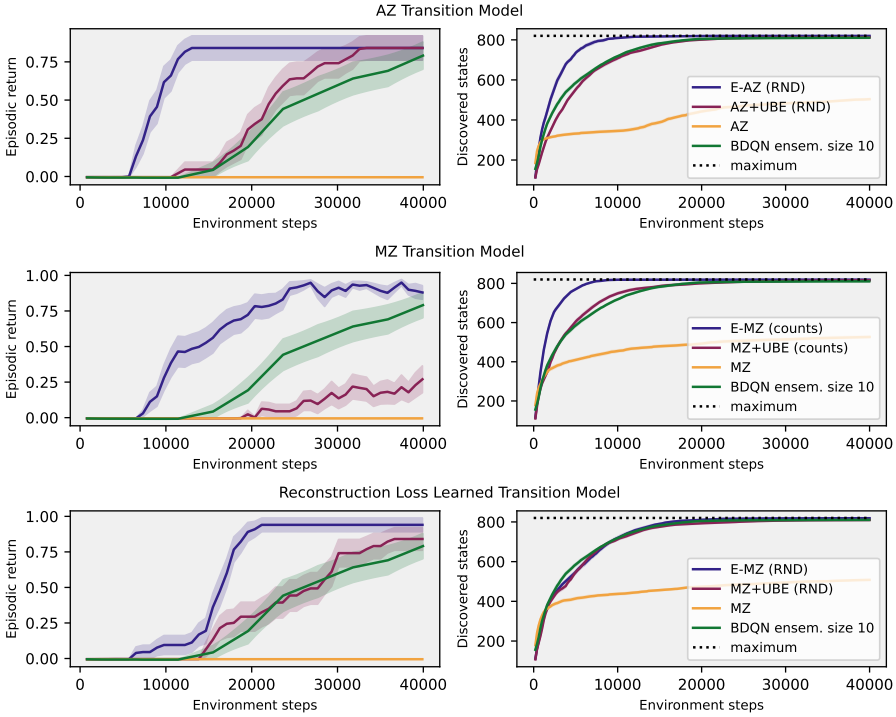

Figure 4: Deep Sea 40x40, mean and standard error for 20 seeds. Rows: Different transition models. Left: episodic return in evaluation vs. environment steps. Right: exploration rate (number of discovered states vs. environment steps).

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

Novelty Estimator	Exploration	Average steps to goal transition for seeds that discovered goal ± STD	% seeds that discovered goal
RND	E-AZ	**10539** ± 9006	94% of 35 seeds
	AZ+UBE	22801 ± 7514	91% of 35 seeds
	AZ	-	0% of 20 seeds
Counts	E-MZ	**14339** ± 6845	100% of 23 seeds
	MZ+UBE	29945 ± 8113	57% of 21 seeds
	MZ	-	0% of 20 seeds
Reconstruction Model (RND)	E-MZ	**15241** ± 3236	95% of 20 seeds
	MZ+UBE	22497 ± 6645	85% of 20 seeds
	MZ	-	0% of 20 seeds

C SUBLEQ

What follows is a formal characterization of SUBLEQ as used in our experiments, and example programs for the two tasks we presented.

For SUBLEQ-N, we have a *memory* made up of N *words* $(w_0, w_1, ..., w_{N-1})$ where each word w_i is an integer from 0 to $N-1$ inclusive (we write $w_i \in [0, N-1]$). We also have an input sequence $(v_0, v_1, ...)$, $v_i \in [0, N-1]$, of variable finite length depending on the particular task to be solved. Tasks correspond to different algorithms we want the agent to implement. Lastly, there is an output buffer $(u_0, u_1, ...)$, $u_i \in [0, N-1]$ which begins empty, but can be extended with output values during execution of a SUBLEQ program.

We refer to w_i as the word at *address* i. Additionally, we give names to some specific addresses: `@IN` $= N-3$, `@OUT` $= N-2$, `@HALT` $= N-1$. The program is stored contiguously in memory starting at address 0. Execution begins from address 0 as well, meaning the first instruction to execute is (w_0, w_1, w_2).

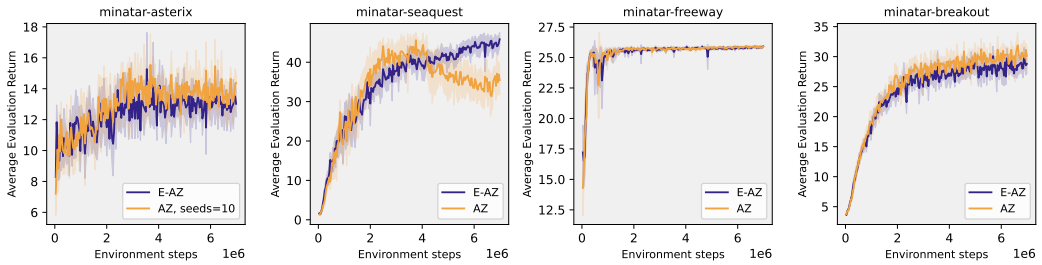

Figure 5: Mean and two standard errors for 10 seeds.

When executing some instruction at address i, we look at the three words (w_i, w_{i+1}, w_{i+2}), and modify the memory by subtracting the value at address w_{i+1} from the word at address w_i, i.e. $w_{w_i} \leftarrow w_{w_i} - w_{w_{i+1}}$. We say that this instruction *reads* from w_i and w_{i+1}, and *writes* to w_i. We also say that reading from w_i *returns* w_{w_i}. All operations are done modulo N so that the words remain in the range $[0, N-1]$. If the result before modulo was less or equal to zero, execution continues from address w_{i+2} (we say we *jump* to w_{i+2}), meaning the next instruction would be $(w_{w_{i+2}}, w_{w_{i+2}+1}, w_{w_{i+2}+2})$. If the result before modulo was strictly positive, then the next instruction is the next three words, i.e. $(w_{i+3}, w_{i+4}, w_{i+5})$.

Interacting with the addresses `@IN`, `@OUT`, and `@HALT` has different behaviour than normal execution. Reading from the address `@IN` will instead read the next number from the input sequence. The first read from `@IN` will return v_0, the next read returns v_1, etc. Reading from the input sequence when there is no next value will return 0 instead. Writing to `@IN` is ignored.

Reading from `@OUT` always returns a 0, but writing to `@OUT` will write to the output buffer. We say that we *output* the value. Each value that we output is added to the output buffer, so after the first output u_0, the buffer looks like (u_0), after the second output u_1, it is (u_0, u_1), etc.

The output buffer is compared against the desired output for the task to determine whether the task was solved correctly. The first incorrect output terminates the program and results in a failure of the task. If the output buffer ever equals the desired task output, the program terminates and the task is solved successfully. Finally, if the program tries jumping to an address where the instruction would overlap `@HALT`, such as $(w_{\texttt{@IN}}, w_{\texttt{@OUT}}, w_{\texttt{@HALT}})$, the program terminates.

When using SUBLEQ-N as a reinforcement learning environment, the agent writes a program one word at a time. It gets to observe the current state of the memory before execution (it is just the program it has written up to that point followed by zeroes), an example input and desired output pair for the given task, and the resulting state of the input sequence and the output buffer after executing the currently written program.

At each state, the agent has N actions $= \{0, 1, ..., N-1\}$, each corresponding to writing that number into the next location in memory. The memory is initially filled with zeroes. After an action is chosen, the number is added to memory, and the currently written program is executed to determine the states of the input and output on the example test as well as other (hidden) test cases. If all test cases succeed, the episode terminates after one more (irrelevant) action. On the other hand, if the agent reaches the end of writable memory ($N-3$ actions), the episode terminates unsuccessfully.

The Negate Positives task expects a program which receives positive integers and outputs that input, expect negated. This task are very simple, since it can be done with a single instruction: Reading from `@IN` will subtract the next input from whichever address to which we write. If we write to `@OUT`, which reads as 0, we are effectively writing $0 - v_i = -v_i$. For Negate Positives, this is exactly what is required, so the only remaining challenge is how to loop back to the start. Luckily, since we know the input is always positive, the output will be negative (or zero), so we always jump. Thus, if we choose the jump address to be 0, we can also loop to the start in the same instruction.

(@OUT, @IN, 0, ...) in memory forms a solution to the task, and since the memory is initialized with zeroes, it only requires the agent to write 2 words, meaning only 2 actions.

In code, a solution to Negate Positives looks like this:

```
@start:
subleq @OUT @IN @start
```

Which the agent would write as:

```
(N-2, N-3, )
```

In the Identity task, we require a program which outputs the input unchanged. This is slightly more difficult that Negate Positives, because an instruction in SUBLEQ always subtracts, so we need at least two instruction to undo the negation. This can be done by storing the return of @IN in some auxiliary word, and then writing that value to output. Let us denote the address at which we store the value temporarily as @x. The program that solves Identity could begin (@x, @IN, 3, @OUT, @x, ...). Note the 3 at address 2 is needed to progress to the second instruction regardless of whether the input is positive or negative. This program would work for the first input, but we must produce a program which loops. Thus, we still need to erase the value stored at address @x (to prepare it for the next write), and we must loop to the start. These two things can be done in the same instruction. We arrive at a program like (@x, @IN, 3, @OUT, @x, 6, @x, @x, 0, ...). The last instruction (@x, @x, 0) clears @x and jumps to address zero (since $w_{@x} \leftarrow w_{@x} - w_{@x} = 0$ which ≤ 0). Again, we required the constant 6 at address 5 to unconditionally continue to the next address. In general, an agent would need 8 actions to solve Identity, where it needs to pick @IN, @OUT, 3, and 6 specifically, and it needs to make sure that all of @x are the same. We say in general, because there is actually a clever solution when we choose $@x = 0$. In that case, the program becomes (0, @IN, 3, @OUT, 0, 6, 0, 0, 0, ...) which only requires the agent to write 6 words, since the memory is initially filled with zeroes.

In code, a general solution to Identity looks like this (left in human syntax, right as the agent writes, the 9s can be any constant $\geq 9, < N$):

```
@start:
subleq @x @IN          ; (9, N-3, 3)
subleq @OUT @x         ; (N-2, 9, 6)
subleq @x @x @start    ; (9, 9, )
@x: .data 0            ; (, ...)
```

While the shortest (known) solution (in terms of non-zero bytes written) looks like this:

```
@start:
subleq @x:@x @IN       ; (0, N-3, 3)
subleq @OUT @x         ; (N-2, 0, 6)
subleq @x @x @start    ; (, , )
```

D Implementation Details for Deep Sea

Our implementation for the Deep Sea agents is based in the framework of Ye et al. (2021). Below, we detail the implementation details unique to E-A/MZ and A/MZ+UBE in Deep Sea.

D.1 Targets

Value, policy and reward targets were all computed as in MZ Schrittwieser et al. (2020). UBE targets were computed in an n-step manner:

$$u_{target}(s_t) = \sum_{i=0}^{n-1} \gamma^{2i} \eta(s_{t+i}, a_{t+i}) + \gamma^{2n} \mathbb{V}[\hat{V}_{\hat{M}}^{\pi}(s_{t+n})] \tag{22}$$

Where η is the RND / exact or hash count-based novelty estimators. To guarantee that the UBE estimates remains sufficiently optimistic, the value-uncertainty bootstrap $\mathbb{V}[\hat{V}^{\pi}_{\hat{M}}(s_{t+n})]$ was computed in one of two ways:

1. *Root targets*: The uncertainty at the root of the EMCTS tree at state s_{t+n}.
2. *Non-root targets*:

$$\mathbb{V}[\hat{V}^{\pi}_{\hat{M}}(s_{t+n})] = \max_{a} \eta(s_{t+n}, a_{t+n}) + \gamma^2 u(s_{t+n+1}), \tag{23}$$

Where

$$u(s_{t+n+1}) = \max\left(\hat{u}(s_{t+n+1}), \frac{1}{1-\gamma^2}\mathbb{V}[\hat{R}(s_{t+n}, a_{t+n})]\right) \tag{24}$$

D.2 Losses

The original MZ algorithm uses three loss functions L_r, L_v, L_π for the reward, value and policy, respectively. The gradients from the losses $\mathcal{L}_r, \mathcal{L}_v, \mathcal{L}_\pi$ propagate through the learned 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:

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

where $\mathcal{B} \equiv \{s_t, a_t, r_t, s_{t+1}, a_{t+1}, \ldots, s_{t+l}\}_{t\in\mathcal{B}}$ is a training batch containing b trajectories of length l sampled from different episodes, and $\hat{s}^k_t$ is the state predicted by the learned model. To 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.

To estimate value-uncertainty at the leaves, we train a UBE function u with a UBE loss $\mathcal{L}_u$:

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

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.

D.3 Different Dynamics Models

We describe the three transition models used in 5 in more detail. The AlphaZero dynamics model is a true model of the dynamics of the environment, in the true state space of the environment. When planning with this model local uncertainty is estimated with RND and value-uncertainty is estimated with UBE. The MuZero model is a value-equivalent model in latent space. g, f are learned by the agent during training from the value, policy, reward and UBE losses. When planning with this model local uncertainty is estimated with state-visitation-counts (see D.5 and value-uncertainty is estimated with UBE. The anchored-MuZero transition model trained only to predict the true transition dynamics of the environment through a reconstruction loss L^k_{re} (see Appendix D.2). When planning with this model local uncertainty is estimated with RND and value-uncertainty is estimated with UBE.

D.4 Planning with Random Network Distillation Based Epistemic Uncertainty

In order to estimate the epistemic uncertainty of a transition $\eta(s, a)$, RND (Burda et al., 2019) take as input state s and action a and computes $L_2(\phi(s, a), \phi'(s, a))$ between two neural networks ϕ, ϕ'. ϕ' is kept stationary, while ϕ is trained with the same loss that evaluates the uncertainty. When the planning is done with a true model, the agent has access to the true states s_{t+k}. 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}.

D.5 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 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 formula:

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

Where $\epsilon > 0$ 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 $\eta(s_{t+k}, a_{t+k}) \to 0$ quickly in observed transitions) relies entirely on the learned UBE function u which operates directly on latent states $\hat{s}_t^k$.

D.6 Using UBE to Estimate Value-Uncertainty at the Leaves

It is essential for exploration that the epistemic uncertainty prediction is reliably high in unobserved areas of the state action space. For this reason, a learned function $\hat{u} \approx u$ may not be sufficient to detect that a state s_t has not been previously observed. Instead, we use the following:

$$\max\left(\hat{u}(s_t), \frac{1}{1-\gamma^2}\eta(s_t, \pi(s_t))\right) \tag{25}$$

If the uncertainty $\eta(s_t, \pi(s_t))$ for the transition $(s_t, \pi(s_t))$ is high, the uncertainty will be estimated as high regardless of the UBE prediction $u(s_t)$, and otherwise, either $u(s_t)$ is high or both are negligible.

D.7 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 to cause instability in training even in off-policy agents. To mitigate that, the EMCTS 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 EMCTS 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. To avoid learning a a separate prior-policy that may not be necessary in environments with small actions space, we set the policy prediction $\pi(s_k)$ (see Equation 3) to uniform over all actions, for all s_k during exploration episodes. Dirichlet noise was not used to drive exploration in MCTS with the UBE and EMCTS agents, as any little amount of stochasticity in the policy can prevent the agent from reliably completing the one optimal trajectory.

D.8 The A/MZ+UBE Agent

The A/MZ+UBE ablation agent uses MCTS to evaluate the q value of actions in the same manner as A/MZ, and explores by taking the action a_t that maximizes the combination of the Q-values approximated by MCTS, local uncertainty $\mathbb{V}[\hat{R}]$ and UBE head u:

$$a_t \;=\; \arg\max_a q_{\hat{M}}(s_0, a_t) + \beta\sqrt{\mathbb{V}[\hat{R}(s_0, a_t)] + \gamma^2 u(f(s_0, a_t))}. \tag{26}$$

$q_{\hat{M}}(s_0, a_t)$ are the values at the root of the regular MCTS tree after search. The main difference between A/MZ+UBE and E-A/MZ is that in E-A/MZ the uncertainty $\sqrt{\mathbb{V}[\hat{R}(s_0, a_t)] + \gamma^2 u(f(s_0, a_t))}$ estimated takes into account estimates from multiple different future trajectories, in the manner MCTS estimates the values $q_{\hat{M}}$.

E Network Architecture & Hyperparameters

E.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) for Deep Sea and Koyamada et al. (2023) for SUBLEQ with the objective of achieving learning stability on the target environment with the simplest network architectures 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 in Deep Sea was tuned with the objective that the EMCTS 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.

For SUBLEQ we chose $\beta = 1$. We did not experiment with additional values of β. However, first, the values, UBE prediction, and state-novelty predictions are all bounded ≤ 1, such that β need not account in this case to the possibly arbitrary scales of UBE / the novelty estimator. Second, to make the most out of Jax's naturally parallelized setup, each parallel episode explores with a different $\beta_i \leq \beta$, evenly spaced from 0 to β.

E.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 for Deep Sea. For SUBLEQ the FC network architecture constituted value, UBE, exploration policy π_e and exploitation policy π networks. All networks used two hidden layers of size 256 with ReLu activations between hidden layers. The value head used tanh activation on the last layer, the UBE head used a $0.5(tanh(x) + 1)$ activation to bound the ube prediction between 0 and 1. The policy heads did not use any activation layers.

Table 2: Network architecture hyperparameters, Deep Sea

	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

E.3 Deep Sea Hyperparameter Configuration

We detail the full set of hyperparameters in Tables 3 and 4 for Deep Sea. 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 EMCTS: number of unroll steps times batch size $5 \cdot 256 = 1230$. Otherwise, the default hyper parameters were used.

E.4 SUBLEQ Hyperparameter Configuration

The SUBLEQ E-/AZ agents are implemented in Jax, based in the implementation of Koyamada et al. (2023). MCTS parameters used the defaults provided by DeepMind et al. (2020). Detailed hyperparameter configuration below:

```
hash: XXHash, 32bit
number of parallel episodes = 128
number of E/MCTS simulations = 32
batch size = 4096
number of times each frame appears in training in expectation = 4
discount = 0.97
replay buffer size = 200,000
priority exponent of prioritized replay buffer = 0.6
learning rate = 0.001
populate replay buffer for N frames before starting training, N = 5000
run evaluation episode every N frames, N = 20480
number of parallel evaluation episodes = 32
```

Table 3: Hyperparameters used in the Deep Sea experiments

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 & 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)	$0.4 \rightarrow 1$	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		
	EMCTS	UBE	Uninf.	EMCTS	UBE	Uninf.	EMCTS	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