A. Policy Evaluation

Let $\mathrm{IG}^*(\pi)$ be the information gain of a policy π given true dynamics:

$$\begin{aligned}\mathrm{IG}^*(\pi) &= \mathbb{E}_{\phi\sim\mathcal{P}(\Phi|\pi)}\left[\mathrm{IG}^*(\phi)\right], \\ &= \int_\Phi \mathrm{IG}^*(\phi) p(\phi|\pi, t^*)\, d\phi,\end{aligned}$$

where $p(\phi|\pi, t^*) = p(s, a, s'|\pi, t^*)$ is the probability of transition (s, a, s') occurring given π is the behavioural policy in the external MDP with transition function t^*. Since t^* is unknown, the utility of a policy has to be estimated with over our belief $\mathcal{P}(T)$. Let $\bar{t}$ be the effective expected transition function based on our belief such that:

$$p(\phi|\bar{t}) := \int_T p(\phi|t)p(t)dt, \tag{11}$$

$$= \mathop{\mathbb{E}}_{t\sim\mathcal{P}(T)}\left[p(\phi|t)\right]. \tag{12}$$

Hence, information gain given the dynamics over our belief, $\mathrm{IG}(\pi)$ is:

$$\mathrm{IG}(\pi) = \int_\Phi \mathrm{IG}(\phi) p(\phi|\pi, \bar{t})\, d\phi,$$

which can be expanded as:

$$\mathrm{IG}(\pi) = \int_\mathcal{S}\int_\mathcal{A}\int_\mathcal{S} \mathrm{IG}(s, a, s') p(s, a, s'|\pi, \bar{t})\, ds'\, da\, ds,$$

or:

$$\mathrm{IG}(\pi) = \int_\mathcal{S}\int_\mathcal{A}\int_\mathcal{S} \mathrm{IG}(s, a, s') p(s'|s, a, \bar{t}) p(s, a|\pi, \bar{t})\, ds'\, da\, ds \tag{13}$$

Define action utility function $u(s, a)$ (Equation 4) which quantifies the net utility of taking action a from state s as:

$$u(s, a) = \int_T\int_\mathcal{S} \mathrm{IG}(s, a, s') p(s'|s, a, t)\, ds'\, dt, \tag{14}$$

which is the same as:

$$u(s, a) = \int_\mathcal{S} \mathrm{IG}(s, a, s') p(s'|s, a, \bar{t})\, ds'.$$

using which, Equation 13 can be rewritten as

$$\mathrm{IG}(\pi) = \int_\mathcal{S}\int_\mathcal{A} u(s, a) p(s, a|\pi, \bar{t})\, da\, ds$$

which reduces to the following expectation (Equation 15):

$$\mathrm{IG}(\pi) = \mathbb{E}_{t\sim\mathcal{P}(T)}\left[\mathbb{E}_{s,a\sim\mathcal{P}(\mathcal{S},\mathcal{A}|\pi,T)}\left[u(s, a)\right]\right] \tag{15}$$

B. Action Evaluation

To obtain a closed form expression for $u(s, a)$, Equation 4 can be expanded using Equation 1 as:

$$\begin{aligned}u(s, a) &= \int_\mathcal{S} \mathrm{IG}(s, a, s') p(s'|a, s, \bar{t})\, ds', \\ &= \int_\mathcal{S} D_{\mathrm{KL}}(\mathcal{P}(T|\phi) \parallel \mathcal{P}(T))\, p(s'|s, a, \bar{t})\, ds'\end{aligned}$$

Expanding the equation from definition of KL-divergence:

$$D_{\mathrm{KL}}(\mathcal{P}_1 \parallel \mathcal{P}_2) = \int_Z p_1(z) \log\left(\frac{p_1(z)}{p_2(z)}\right) dz$$

$$u(s,a) = \int_{\mathcal{S}} \int_T p(t|\phi) \log\left(\frac{p(t|\phi)}{p(t)}\right) p(s'|s,a,\bar{t})\, dt\, ds' \tag{16}$$

The prior $p(T)$ and the posterior $p(T|\phi)$ are related by Bayes rule:

$$p(T|\Phi) = \frac{p(\Phi|T)p(T)}{p(\Phi)} \tag{17}$$

Equation 16 can be now rewritten as:

$$u(s,a) = \int_{\mathcal{S}} \int_T \frac{p(\phi|t)p(t)}{p(\phi)} \log\left(\frac{p(\phi|t)}{p(\phi)}\right) p(s'|s,a,\bar{t})\, dt\, ds'$$

$p(\phi)$ is actually $p(\phi|D)$ where D is the history of agent comprising of the data it has collected so far about the environment. Hence $p(\phi)$ can be computed as the marginal over our belief of transition functions:

$$\begin{aligned} p(\phi|D) &= p(\phi|T), \\ &= \int_T p(\phi|t)p(t)\, dt, \\ &= p(\phi|\bar{t}), \end{aligned}$$

therefore,

$$u(s,a) = \int_{\mathcal{S}} \int_T \frac{p(\phi|t)p(t)}{p(\phi|\bar{t})} \log\left(\frac{p(\phi|t)}{p(\phi|\bar{t})}\right) p(s'|s,a,\bar{t})\, dt\, ds' \tag{18}$$

Substituting $\phi = (s,a,s')$ and since s,a are given:

$$u(s,a) = \int_{\mathcal{S}} \int_T \frac{p(s,a,s'|t)p(t)}{p(s,a,s'|\bar{t})} \log\left(\frac{p(s,a,s'|t)}{p(s,a,s'|\bar{t})}\right) p(s'|s,a,\bar{t})\, dt\, ds'$$

$$u(s,a) = \int_{\mathcal{S}} \int_T p(s'|s,a,t)p(t) \log\left(\frac{p(s'|s,a,t)}{p(s'|s,a,\bar{t})}\right) dt\, ds' \tag{19}$$

Expanding and swapping integrals in the former term:

$$\begin{aligned} u(s,a) = &\int_T \int_{\mathcal{S}} p(s'|s,a,t) \log\left(p(s'|s,a,t)\right) p(t)\, ds'\, dt \\ - &\int_{\mathcal{S}} \int_T p(s'|s,a,t)p(t)\, dt \log\left(\int_T p(s'|s,a,t)p(t)\, dt\right) ds' \end{aligned} \tag{20}$$

$-\int_z p(z) \log\left(p(z)\right) dz$ is just entropy $\mathfrak{H}(\mathcal{P}(z))$.

$$u(s,a) = \mathfrak{H}\left(\int_T \mathcal{P}(\mathcal{S}|s,a,t)p(t)\, dt\right) - \int_T \mathfrak{H}(\mathcal{P}(\mathcal{S}|s,a,t))p(t)\, dt \tag{21}$$

$$= \mathfrak{H}\left(\mathbb{E}_{t\sim\mathcal{P}(T)}\left[\mathcal{P}(\mathcal{S}|s,a,t)\right]\right) - \mathbb{E}_{t\sim\mathcal{P}(T)}\left[\mathfrak{H}(\mathcal{P}(\mathcal{S}|s,a,t))\right] \tag{22}$$

$\mathcal{P}(\mathcal{S}|s,a,t)$ represents a probability distribution over the next state with removal of the integrals over s'. Given a space of distributions, the entropy of the average distribution minus the average entropy of distributions is the Jensen Shannon Divergence (JSD) of the space of distributions. It is also termed as the Information Radius and is defined as:

$$\text{JSD}\{\mathcal{P}_{\bar{Z}} \mid \mathcal{P}_{\bar{Z}} \sim \mathcal{P}(\mathbf{Z})\} = \mathfrak{H}\left(\int_{\bar{Z}} \mathcal{P}_{\bar{Z}} p(\mathcal{P}_{\bar{Z}})\, d\bar{Z}\right) - \int_{\bar{Z}} \mathfrak{H}(\mathcal{P}_{\bar{Z}}) p(\mathcal{P}_{\bar{Z}})\, d\bar{Z} \tag{23}$$

where $\mathbf{Z}$ is the space of distributions $\mathcal{P}_{\bar{Z}}$. $\mathcal{P}(\mathbf{Z})$ is a compound probability distribution with $p(\mathcal{P}_{\bar{Z}})$ as its density function.

Therefore, Equation 14 can be expressed as:

$$u(s,a) = \text{JSD}\{\mathcal{P}(\mathcal{S}|s,a,t) \mid t \sim \mathcal{P}(T)\} \tag{24}$$

where T is the space of transition functions with probabilistic outputs.

C. Chain Environment

C.1. Stochastic Environment

The left-most state (state 0) of the Chain Environment was modified to be a *stochastic trap* state. If the agent is in the trap state, regardless of the chosen action, it is equally likely to remain in state 0 or end up in state 1. A method that relies only on prediction errors cannot separate risk from uncertainty. Hence it would repeatedly visit the trap state as it is easier to reach it compared to a far-off unexplored state. Figure 1d compares the performance of our method on both the normal Chain environment and the one with a stochastic trap state. Although MAX is slowed down, it still manages to explore the transitions. The stochastic trap state increases the variance in the output of the models. This is because the models were trained on different outputs for the same input. Hence the resulting output distribution of the models were sensitive to the training samples the models have seen. This can cause disagreements and hence result in a higher utility right next to the initial state. The true uncertainty on the right, which is fainter, is no longer the dominant source of uncertainty. This causes even our method to struggle, despite being able to separate the two forms of uncertainty.

C.2. MAX

The ensemble consisted of 3 forward models implemented as fully-connected neural networks, each receiving one-hot-encoded state of the environment and the action of the agent as input. The outputs of the network were categorical distributions over the (next) states. The networks were independently and randomly initialized which was found to be enough to foster the diversity for the out-of-sample predictions and hence bootstrap sampling was not used. After 3-episode warm-up with random actions, the models were trained for 150 iterations. Searching for an exploration policy in the exploration MDP, which was an open-loop plan, was performed using 25 rounds of Monte Carlo Tree Search (MCTS), which used 5 random trajectories for each step of expansion with Thompson Sampling as the selection policy. The best action was chosen based on the average value of the children. Models were trained after experiencing each transition in the environment and the MCTS tree was discarded after every step. The hyper-parameters of MAX were tuned with grid search for the chain length of $N = 50$.

Table 1. Hyper-Parameters for MAX. *Grid Size: 0.8k*

Hyper-parameter	**Values**			
Hidden Layers	2	3	4	
Hidden Layer Size	64	128	256	
Learning Rate	10^{-3}	10^{-4}		
Batch Size	16	64	256	
Training Iterations per Episode	16	32	64	128
Weight Decay	0	10^{-5}	10^{-6}	10^{-7}

C.3. Baselines

Exploration Bonus DQN is an extension of the DQN algorithm (Mnih et al., 2015), in which the transitions that are visited for the first time carry an extra bonus reward. This causes the value of those transitions to be temporarily over-estimated

and results in more visits to novel regions of the environment. In general, the exploration bonuses can be computed using prediction errors of forward models (Pathak et al., 2017). However, for our simple environment, the bonuses were provided by an oracle, implemented as a transition look-up table. This is equivalent to having an ideal model that can learn about a transition in one-shot, therefore emulating the best-case scenario for the method.

Bootstrapped DQN by (Osband et al., 2016) also extends the DQN algorithm and it is based on the same underlying principle as Exploration Bonus DQN. Instead of introducing arbitrary extra rewards, it relies on stochastic over-estimation of values in novel states. Multiple Q-value networks or *heads* are maintained and trained on a shared replay buffer. Before every episode, a head is randomly selected and the agent acts greedily with respect to it. If a novel transition (s, a, s') is added to the replay buffer, the hope is that at least one of the heads over-estimates the value of some action a' in state s or s', causing the policy resulting from TD-bootstrapping to prefer this state. This leads to that particular state being further explored when that head is used to act.

Table 2. Hyper-Parameters for DQN with Exploration Bonus. *Grid Size: 4.3k*

Hyper-parameter	**Values**		
Exploration Bonus	5×10^{-3}	10^{-2}	10^{-1}
Replay Size	256	10^5	
Hidden Layers	1	2	3
Hidden Layer Size	32	64	128
Learning Rate	10^{-2}	10^{-3}	10^{-4}
Batch Size	16	32	64
Discount Factor	0.95	0.975	0.99
Target Network Update Frequency	16	64	256

Table 3. Hyper-Parameters for Bootstrapped DQN. *Grid Size: 2.1k*

Hyper-parameter	**Values**		
Hidden Layers	1	2	3
Hidden Layer Size	32	64	128
Learning Rate	10^{-2}	10^{-3}	10^{-4}
Batch Size	16	32	64
Training Iterations per Episode	16	32	64
Discount Factor	0.95	0.975	0.99
Target Network Update Frequency	16	64	256

Commonly used ϵ-greedy exploration was turned off in both baselines, thereby making exploration reliant solely on the methods themselves. The Q-value functions were implemented with multi-layer fully connected neural networks. The state was encoded with thermometer encoding for Bootstrapped DQN as proposed by Osband et al. (2016). The Q-value network in Exploration Bonus DQN was trained with one batch of data after each step in the environment. For Bootstrapped DQN, 10 Q-value network heads were trained *only after* each episode, with the exact number of iterations being tuned. Bootstrap sampling was not employed and all heads were trained on all transitions collected so far. Target networks and replays were used for both methods as in the standard DQN. RMSprop optimizer with a momentum of 0.9 was used to minimize the Huber Loss with gradient clipping of 5.

All neural networks consisted of Glorot-initialized, *tanh*-activated fully connected layers with their depth and width being tuned.

Each hyper-parameter configuration was tested using 5 different random seeds. Hyper-parameters were ranked according to the median of the area under the exploration curve.

C.4. Supplementary Figures

Figures 5a, 5b and 6 analyze the behaviour of the proposed algorithm. Fig. 5a plots the utility of the exploration policy during two learning episodes. Clearly, high utility correlates with experiencing novel transitions. Figure 5b, which shows the learning curves of individual models in an ensemble, indicates that less than 20 episodes is required to correctly learn all the transitions. The surrogate models were therefore indistinguishable from the environment. Notice also that the learning curves of the three models are close to each other. This is because the models nearly always agree on the transitions from the training set and disagree on the others. The exploration is thus mainly driven by the divergence of predictions for the unobserved transitions. Figure 6 visualizes the transitions in the final two episodes of a run showing how MAX plans for uncertain (unvisited) transitions, which results in minimizing their corresponding uncertainties.

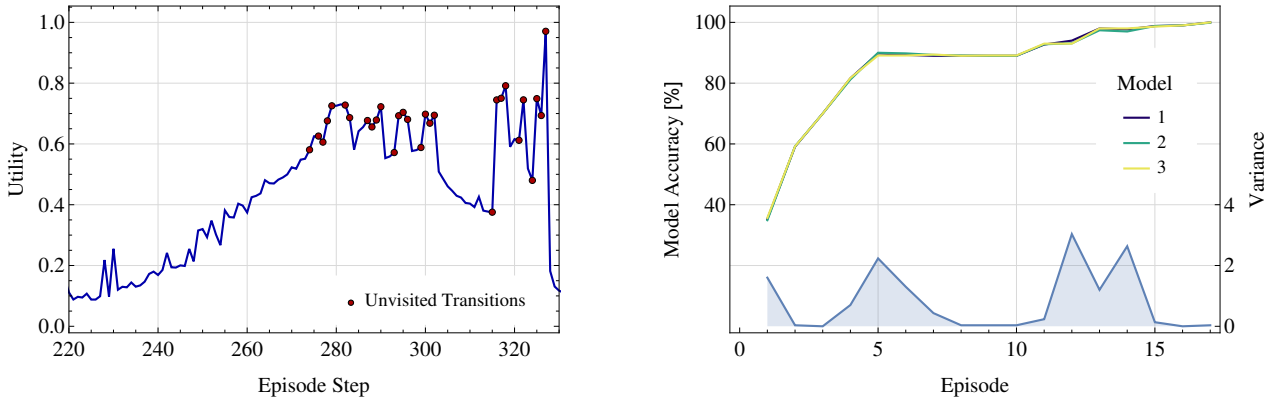

Figure 5. Exploration with MAX. (a) Utility of the exploration policy (in normalized log scale) during an exemplary episode. The red points mark encounters with novel transitions. (b) The accuracy of the models in the ensemble at each episode and the variance of model accuracy.

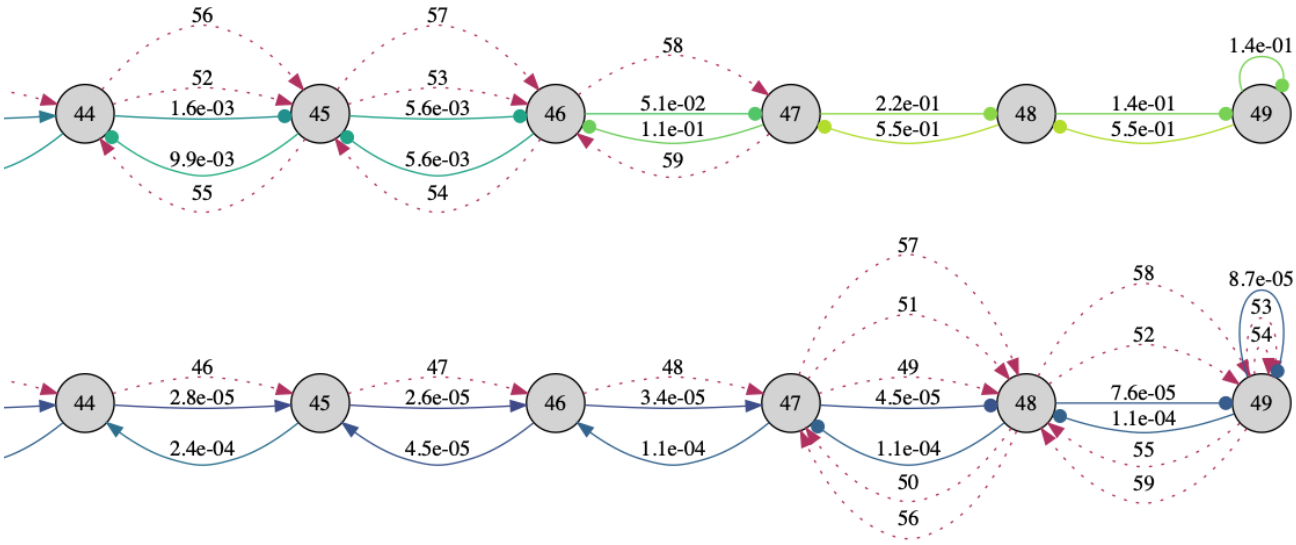

Figure 6. Instance of uncertainty minimization:MAX behaviour in the second-to-last (top) and the last (bottom) episodes. The solid arrows (environment transitions) and are annotated with the utilities of predicted information gains. The arrowheads are circular for the unvisited transitions and regular for the visited ones (before an episode). The dotted arrows with step numbers visualize the path the agent was following during an episode (only the final steps are shown).

Robustness to Key Hyper-Parameters Number of models in the ensemble, trajectories per an MCTS iteration, and MCTS iterations were varied and the results are shown in Figure 7. In general, more trajectories, more models in the ensemble and more planning iterations lead to better performance. Notice, however, that the algorithms works efficiently also for minimal settings.

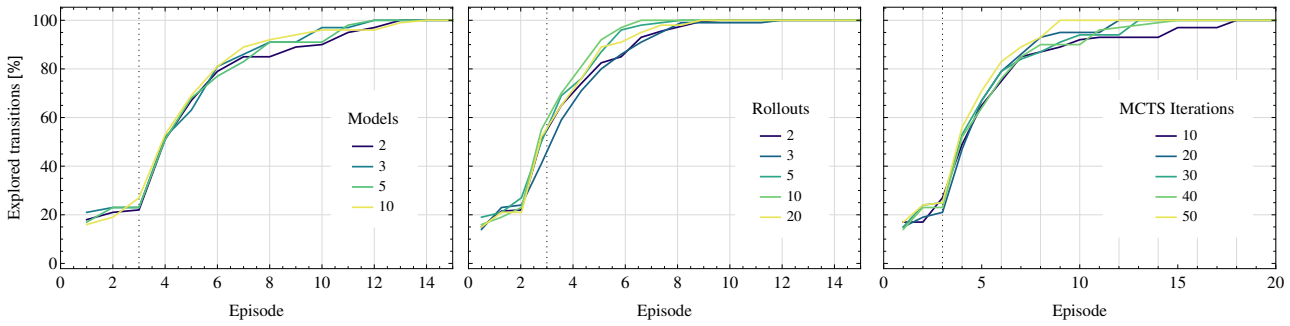

Figure 7. Algorithm Properties. Each learning curve is a median of 5 runs with different seeds. Vertical dotted line marks the ends of the warm-up phase. Sub-figures show how the percentage of explored transitions varies with respect to (a) ensemble size, (b) number of rollouts and (c) planning iterations.

D. Continuous Environments

D.1. Numerically Stable Jensen-Rényi Divergence Implementation

$$\mathfrak{D}(\mathcal{N}_i, \mathcal{N}_j) = \frac{1}{|\Sigma_i + \Sigma_j|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\left(\mu_j - \mu_i\right)^T \left(\Sigma_i + \Sigma_j\right)^{-1} \left(\mu_j - \mu_i\right)\right)$$

$$\begin{aligned} p &= \left(u_j - u_i\right)^T \left(\Sigma_i + \Sigma_j\right)^{-1} \left(\mu_j - \mu_i\right) \\ q &= \ln(|\Sigma_i + \Sigma_j|) \\ &= \operatorname{trace}(\ln(\Sigma_i + \Sigma_j)) \end{aligned}$$

$$\mathfrak{D}(\mathcal{N}_i, \mathcal{N}_j) = \exp\left(-\frac{1}{2}(p+q)\right)$$

Additionally, LOG-SUM-EXP trick is used to calculate the entropy of the mean.

D.2. Experimental Setup

256 steps of warm-up was used for all methods to train the models before exploration began. During exploration, for all methods, models were retrained from scratch for 50 epochs every 25 steps. SAC policies were also relearned from scratch every 25 steps to avoid over-commitment. The current exploration experience was transferred to SAC as its off-policy data and 100 steps of SAC off-policy training was done. For active methods, there was additional on-policy training done using data generated during 50 episodes, each of length 50, starting from current state using 128 actors and the model ensemble. The SAC entropy weight α was tuned separately for each method. It was found to be 0.02 for Rényi entropy based methods MAX and JDRX, and 0.2 for the TVAX and PERX.

Exploitation in Half Cheetah was done every 2000 steps of exploration. Using the exploration data, the model ensemble was trained for 200 epochs. For all methods, SAC policies were trained using both off-policy data from exploration and with on-policy data from the model ensemble. On-policy data was generated using 250 episodes of length 100 using 128 actors. Note that this is recursive prediction for 100 steps starting from the start state. As with exploration, the next states

Table 4. Hyper-Parameters for Models in Continuous Environments

Hyper-parameter	Value
Ensemble Size	32
Hidden Layers	4
Hidden Layer Size	512
Batch Size	256
Non-linearity	Swish
Learning Rate	0.001

were sampled at each transition by randomly selecting a model from the ensemble and then sampling the next state from its output distribution. A policy was trained and evaluated 3 times and the average performance was reported.

Models were trained to predict the state deltas, instead of raw next states as is common. The states, actions and deltas were all normalized to have zero mean and unit variance. For all methods, the utility computation was done in normalized scales for stability.

Half Cheetah Reward Functions
Running: $r_t = v_t^x - 0.1\|a_t\|_2^2$; flipping: $r_t = \omega_t^y - 0.1\|a_t\|_2^2$,

where v_t^x is the velocity along the x axis at time t, and ω_t^y is the angular velocity around axis y at time time t.