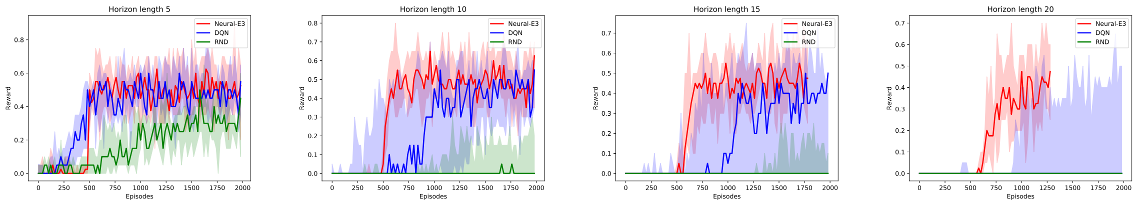

Figure 1: Results on stochastic combination lock environment for different horizon lengths. Solid lines represent median across 5 seeds, shaded regions represent range between best and worst seeds.

Reviewer 1: Thank you for the review. The assumption that $R^\star$ is known is for compactness of notation and does not restrict the class of problems the algorithm can handle - we will clarify this in the paper. For example, consider a setting where states are represented by n-dimensional vectors. We can construct an augmented state space where vectors have $n+1$ dimensions, where the last dimension represents the reward. In this case $R^\star$ simply reads off the last component of the state vector, $R^\star(s) = s[n+1]$, and accurately predicting the reward in the original formulation amounts to accurately predicting the last component of the state vector in this new formulation. In our experiments, we train the model to predict the reward as well as the next state, and do not assume the reward function is known beforehand.

As suggested in the improvements section, we added a new set of experiments in a stochastic environment, the stochastic combination lock in [1]. This is a hard exploration problem where the chance of getting reward with random exploration is exponentially small in horizon length. We will briefly describe the stochastic version of the practical algorithm here, and in detail in the updated paper. The models in the ensemble output the parameters of a distribution over next states rather than single predictions. For each model M we estimate the distribution $P_M^{\pi,h}(\cdot)$ using k points at each step in the rollout, and compute the disagreement-based signal used to drive the exploration by taking the total variation or KL divergence between the empirical distributions for each pair of models. These k points are then used as inputs to the model at the next step in the rollout. We use Monte-Carlo Tree Search as a planner and only execute the first action in the returned sequence, replanning at every step. The results are shown in Figure 1. We see that as the horizon length increases, the performance of the baseline methods degrades whereas Neural-E^3 maintains good performance.

Reviewer 2: Thank you for the review. We will clarify the relationship between the idealized version of the algorithm and the practical version both here and in the paper. For both algorithms, $\mathcal{M}_t$ represents a set of models which are consistent with the experience gathered so far, i.e. have low error on the current replay buffer. For the idealized algorithm, models which have high error are eliminated through the explicit elimination step in Algorithm 2. For the practical algorithm, models which have high error are avoided by the optimization procedure, and are thus are unlikely to be present in $\mathcal{M}_t$. The main difference between the two is that the idealized version maintains *all* models in the model class which have low error, whereas the practical version only maintains a subset due to time/memory constraints.

Both versions of the algorithm are based on two key ideas: i) computing exploration policies designed to induce disagreement between plausible models ii) doing so internally, without having to interact with the environment. The motivation is that every time the agent interacts with the environment, there is a high chance that the experience gathered will be useful for refining the set of models. The theoretical analysis makes this intuition precise, and explains why disagreement between two plausible models implies error with respect to the true model and hence useful experience (this applies to the practical algorithm as well). While the practical algorithm does approximate the full version space by an ensemble, our experiments suggest that this approximation can often be sufficient in practice.

To clarify the point "the author uses the structural properties in factored MDP, which is well-studied": note that our sample complexity result depends on the rank of the misfit matrix, which is indeed low for factored MDPs but may also be low in other settings (for example, it is also low for low-rank MDPs).

Reviewer 3: Thank you for the review. To answer your first question: to get an ϵ-optimal policy, ϕ is set according to the formula in Theorem 1, i.e. $\phi = \frac{\epsilon}{24H^2|\mathcal{A}|^2\sqrt{d}}$. Note that this requires knowledge of the rank of the misfit matrix d, which may not be known beforehand. In this case one can use a doubling trick to estimate d while still maintaining polynomial sample complexity, as in [18, 41]. We will add this in the updated paper. Concerning the extension to infinite action spaces, that is a very interesting research question which we hope to investigate in future work. There has been some work on infinite action spaces in the contextual bandit setting, however extending such results to multi-step RL is to our knowledge still an open problem. We have added experiments for an additional (stochastic) environment as suggested in the improvements section, please see Figure 1 and our reponse to Reviewer 1.

References [1]: *Provably efficient RL with Rich Observations via Latent State Decoding*, Du et. al, (ICML 2019).