EXPLORING BY EXPLOITING BAD MODELS IN MODEL-BASED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Exploration for reinforcement learning (RL) is well-studied for model-free methods but a relatively unexplored topic for model-based methods. In this work, we investigate several exploration techniques injected into the two stages of model-based RL: (1) during optimization: adding transition-space and action-space noise when optimizing a policy using learned dynamics, and (2) after optimization: injecting action-space noise when executing an optimized policy on the real environment. When given a good deterministic dynamics model, like the ground-truth simulation, exploration can significantly improve performance. However, using randomly initialized neural networks to model environment dynamics can *implicitly* induce exploration in model-based RL, reducing the need for explicit exploratory techniques. Surprisingly, we show that in the case of a local optimizer, using a learned model with this implicit exploration can actually *outperform* using the ground-truth model without exploration, while adding exploration to the ground-truth model reduces the performance gap. However, the learned models are highly local, in that they perform well *only* for the task for which it is optimized, and fail to generalize to new targets.

1 INTRODUCTION

Reinforcement learning (RL) with deep neural networks combines two extremely data inefficient techniques, increasing the need for efficient exploration. In model-free methods, exploration is usually done by adding noise to the action suggested by the optimized policy. Here, exploration helps primarily by encouraging the agent to visit new states that might have better reward. On the other hand, model-based RL (MBRL) has been more focused on reaching parity with model-free methods in asymptotic performance on environments with high-dimensional state- and action-spaces (Chua et al. (2018)). As a result, exploration in MBRL is a relatively nascent field, though it is still thought to help by generating diverse environment data to train the dynamics model (Pathak et al. (2019)). However, exploration methods designed for the model-free case may result in non-trivial, potentially problematic, effects when used in MBRL.

In this work, we take a step back and analyze how exploration behavior can be injected into MBRL, and extensively evaluate the effect of various exploration strategies on learning performance. Most MBRL approaches iterate between two phases for learning a task: 1) simulation/planning phase: using the current approximation of the dynamics, optimize a policy which accomplishes the task, and 2) environment interaction phase: the optimized policy is executed in the real world, which allows the agent to collect new observations; the approximate dynamics model is then updated with these new observations.

Exploration can be injected during *both* phases. First, exploration can take place during the real-world execution phase by perturbing the optimized actions, similar to how many model-free methods implement exploration. We call this exploration type *action-space exploration*. Second, exploration can also be performed *during* the planning phase, either by performing action-space exploration during policy optimization itself or by considering the distribution of predicted trajectories, and using a risk-seeking approach to encourage the exploration of parts of the model with high uncertainty. We call this type of exploration *transition-space exploration*.

In this paper, we thoroughly investigate the effect of these different types of exploration strategies on the performance of MBRL. We especially focus our analysis on algorithms that use the iterative Linear Quadratic Regulator (**iLQR**) (Li & Todorov (2004)) as the policy optimization method; a well-established approach both in RL and optimal control settings for real robotic applications (Levine et al. (2016); Bechtle et al. (2019); Viereck et al. (2018); Neunert et al. (2016)). This choice is motivated by our desire to focus on methods that are demonstrably applicable to real robotic motor control problems. We focus our analysis to a reaching task for a 7-DoF arm. Our experimental results lead to the following surprising observations:

- Action-space exploration, especially the commonly-cited iLQR solution for maximum-entropy RL, results in significant deterioration of performance.
- Some exploration methods can improve performance of iLQR with a ground-truth dynamics model, but do not improve performance for iLQR using learned dynamics models.
- In our setting, using learned models with iLQR results in *implicit exploration* that can lead to *better performance* than using a ground-truth dynamics model. In the process of learning a model, iLQR is able to escape local minima despite a poor initialization, while iLQR with a good model is more sensitive to initialization.

We conclude that certain choices for modeling dynamics, such as randomly initialized neural networks, implicitly provide an exploration mechanism in model-based RL and reduce the need and benefit of explicit exploration. However, policy optimization with using a strong, deterministic model, such as the ground-truth dynamics, can benefit from exploration because they lack this implicit exploration mechanism.

2 Related work

2.1 Exploration for model-free RL

Model-free methods have worked exceptionally well using only simple forms of exploration, such as injecting noise in different stages of a greedy (short-term optimal) algorithm. Noise can be injected in the action space for the discrete case through ϵ-greedy exploration (Sutton et al. (1998)) and for the continuous case through additive Gaussian noise or by sampling directly from stochastic policies. Alternatively, noise can be injected into the parameter-space of the policy (Rückstieß et al. (2008); Fortunato et al. (2017); Plappert et al. (2017)), or explore by modeling data from previous tasks (Bogdanovic & Righetti (2019)).

A related line of work also maximizes an auxiliary objective along with the main objective to increase exploration, such as entropy or expected improvement to encourage diverse solutions and sample-efficiency. Pelikan et al. (1999); Snoek et al. (2012) use Bayesian optimization to model the objective landscape and explore by incorporating both the mean and uncertainty about the prediction in an auxiliary optimization. Williams & Peng (1991); Ziebart (2010); Haarnoja et al. (2017; 2018) augment the standard RL objective with an entropy maximization term, which is deeply linked with risk-sensitive optimal control theory (Rawlik et al. (2013)), which we build upon in our work.

2.1.1 Using models for exploration in model-free RL

Curiosity-driven exploration can be summarized as “explore what surprises you”; often, this takes the form of incentivizing exploration of states with high prediction error with respect to a dynamics model (Schmidhuber (1991; 2006)) even while using model-free algorithms to learn to act. This line of work can be scaled up to higher-dimensional environments including images Stadie et al. (2015); Pathak et al. (2017). This may perform well in the absence of extrinsic reward, as Burda et al. (2018a;b) demonstrate, and even when disregarding the temporal structure of RL as in Conti et al. (2018).

2.2 Exploration for model-based RL

There are comparatively fewer exploration methods designed specifically for model-based RL. As mentioned previously, model-based RL methods can perform exploration during the planning phase as well as the system rollout phase; we categorize the following related work based on this distinction.

Exploration during environment interaction: Wiering & Schmidhuber (1998) propose using an exploration Q-function to generate exploratory actions for model-based RL. Lopes et al. (2012) empirically measures learning progress to drive exploratory behavior. Hester & Stone (2017) use random forests for modeling dynamics and encourage exploration using model uncertainty and distance measure from previously explored regions. All of these methods only consider the tabular case, while we examine exploration for model-based RL in high-dimensional continuous state- and action-spaces.

Exploration during planning: Moldovan et al. (2015) use the principle of "optimism in the face of uncertainty" by learning a Dirichlet process mixture of linear models for model-predictive control. Shyam et al. (2018) also use an ensemble of deep neural networks to model forward dynamics and use disagreement between models to provide exploratory signal. Boedecker et al. (2014) use probabilistic dynamics models with iLQR to encourage exploration through model uncertainty; however, they consider lower-dimensional problems using Gaussian Processes, which do not scale well to large number of data points. Our work is most closely related to Bechtle et al. (2019), which uses risk-sensitive stochastic optimal control and the uncertainty of probabilistic dynamics models to provide exploration signal. We follow their formulation but use ensembles of deep neural networks as the dynamics model choice for better scaling properties, simplify the variance estimate used to provide exploration signal, and perform much larger scale simulation experiments.

3 Model-based reinforcement learning (MBRL)

Algorithm 1 MBRL($\boldsymbol{s}_1, F, \hat{F}_\theta, T$, n_iter)

```
1: $\mathcal{D} \leftarrow$ run_policy($\boldsymbol{s}_1, \pi_{\text{babbling}}, F, T$)
2: $\boldsymbol{\theta} \leftarrow$ train_model($\hat{F}_{\boldsymbol{\theta}}, \mathcal{D}$)
3: for iter=1, ..., n_iter do
4:     $\tau_{\text{last}} \leftarrow$ last trajectory from $\mathcal{D}$
5:     $\pi_{\text{iter}} \leftarrow$ derive_policy($\hat{F}_{\boldsymbol{\theta}}, \tau_{\text{last}}, T$)
6:     $\mathcal{D} = \mathcal{D} \cup$ run_policy($\boldsymbol{s}_1, \pi_{\text{iter}}, F, T$)
7:     $\boldsymbol{\theta} \leftarrow$ train_model($\hat{F}_{\boldsymbol{\theta}}, \mathcal{D}$)
8: end for
9: return $\pi_{\text{n\_iter}}, \hat{F}_{\boldsymbol{\theta}}$
```

Algorithm 2 run_policy($\boldsymbol{s}_1, \pi, f, T$)

```
1: $\tau_{\text{new}} = \{\}, s_{t=1} = \boldsymbol{s}_1$
2: for $t = 1, \dots, T$ do
3:     $\boldsymbol{a}_t \sim \pi(\boldsymbol{s}_t)$
4:     $\boldsymbol{s}_{t+1} = f(\boldsymbol{s}, \boldsymbol{a}_t)$
5:     $\tau_{\text{new}} = \tau_{\text{new}} \cup \{(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})\}$
6: end for
7: return $\tau_{\text{new}}$
```

Algorithm 3 derive_policy_iLQR($\hat{F}_{\boldsymbol{\theta}}, \tau_{\text{nominal}}, T$)

```
1: $\mathbb{A} \leftarrow$ line search parameters $[0, \dots, 1]$
2: $\tau^*, J^* \leftarrow \tau_{\text{nom}}$, cost of $(\tau_{\text{nom}})$
3: for opt_iter in $1, \dots,$ max_opt_iters do
4:     $k_t, K_t \leftarrow$ Riccati equation
5:     for $\alpha \in \mathbb{A}$ do
6:         $\tau_{\text{simulated}} \leftarrow$ simulate policy $\hat{\pi}$ using $\hat{F}_{\boldsymbol{\theta}}$
7:         $J_{\text{simulated}} \leftarrow$ cost of $\tau_{\text{simulated}}$
8:         if $J_{\text{simulated}} < J^*$ then
9:             $\tau^*, J^* \leftarrow \tau_{\text{simulated}}, J_{\text{simulated}}$
10:            $\pi^* \leftarrow \hat{\pi}$
11:        end if
12:        if converged then
13:            break
14:        end if
15:    end for
16: end for
17: return $\pi^*$
```

Following Bellman (1957), consider a finite-horizon Markov decision process (MDP) with the time horizon $T \in \mathbb{N}$. In this setting, the aim of reinforcement learning is to find a stochastic policy π which minimizes the expected total cost $C(\pi) = \mathbb{E}_{\boldsymbol{s}_1, \boldsymbol{a}_1, \dots} \sum_{t=1}^{T} c(\boldsymbol{s_t}, \boldsymbol{a_t})$ where $\boldsymbol{s}_1 \sim \rho_1(\boldsymbol{s}_1), \boldsymbol{a}_t \sim \pi(\boldsymbol{a}_t | \boldsymbol{s}_t), \boldsymbol{s}_{t+1} \sim F(\boldsymbol{s}_{t+1} | \boldsymbol{s_t}, \boldsymbol{a_t})$. $\boldsymbol{s}_t, \boldsymbol{a}_t$ is the state and action at time t respectively, and the dynamics function F induces a distribution over the next state $\boldsymbol{s}_{t+1}$.

MBRL (Algorithm 1) approaches this problem by approximating the dynamics function F with a model $\hat{F}$, parameterized by some $\boldsymbol{\theta}$. Given $\hat{F}_{\boldsymbol{\theta}}$, MBRL derives a policy $\pi_{\hat{F}_{\boldsymbol{\theta}}}$ by optimizing the predicted long-term cost. This can be done through sampling-based methods like the cross-entropy method (Chua et al. (2018)) or gradient-based optimal control methods like iterative LQR (Li & Todorov (2004)). Bechtle et al. (2019) show that MBRL with iLQR can be applied to high-dimensional real-world tasks.

3.1 Dynamics model learning through EPNNs

A recently popular choice for modeling dynamics are ensembles of probabilistic neural networks (EPNNs) (Chua et al. (2018)). EPNNs provides a *probabilistic* prediction of the dynamics, and scale well with respect to dataset size. Following Lakshminarayanan et al. (2017): we define a probabilistic neural network (PNN) $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ as a deep neural network which outputs the parameters of a probability distribution, in this case the mean $\boldsymbol{\mu}(\boldsymbol{x})$ and diagonal covariance $\Sigma(\boldsymbol{x})$ of a multivariate Gaussian distribution. The parameters $\boldsymbol{\theta}$ are learned by minimizing the negative log likelihood of data collected on the true dynamics. We define a B-sized ensemble of probabilistic neural networks (EPNN) as the bootstrapped ensemble of B PNN models and treat the output of the ensemble as a uniformly-weighted Gaussian mixture model.

3.2 Policy optimization via iterative Linear Quadratic Regulator (iLQR)

Given a model $\hat{F}$, finite horizon T, a nominal trajectory $\tau^{\text{nom}} = \{(\boldsymbol{s}_1^{\text{nom}}, \boldsymbol{a}_1^{\text{nom}}), \dots, (\boldsymbol{s}_T^{\text{nom}}, \boldsymbol{a}_T^{\text{nom}}\}$, and a known cost function $c(\boldsymbol{s}, \boldsymbol{a})$, iLQR returns a set of control parameters $\boldsymbol{k}_t, K_t$ for a time-varying locally-linear feedback policy: $\boldsymbol{a}_t = \pi_t(\boldsymbol{s}_t) = \boldsymbol{a}_t^{\text{nom}} + \alpha \boldsymbol{k}_t + K_t(\boldsymbol{s}_t^{\text{nom}} - \boldsymbol{s}_t)$, with step size α chosen by line search.

In more detail: given the nominal initial actions $\boldsymbol{a}_t, t = 1, \dots, T$, iLQR in the **forward pass** rolls out the actions using the learned dynamics model $\boldsymbol{s}_{t+1} = \hat{F}(\boldsymbol{s}_t, \boldsymbol{a}_t)$. The policy is then optimized in the **backward pass** by computing a linear approximation of the dynamics and a quadratic approximation of the cost using Riccati equations. We make a linear approximation at each time step $\boldsymbol{s}_{t+1} = \hat{F}_{\boldsymbol{s}_t} \boldsymbol{s}_t + \hat{F}_{\boldsymbol{a}_t} \boldsymbol{a}_t$ to construct the local value function Q_t and its derivatives. This is used to update the parameters of the time-varying locally-linear feedback policies $\boldsymbol{k}_t, K_t$. See Algorithm 3 for details and Li & Todorov (2004) for a derivation.

4 Exploration in model-based reinforcement learning

We consider two dimensions of exploration in MBRL. We first note that we can apply exploration in both the *environment-interaction phase* (when collecting data on the real environment), and during the *simulation phase* (when simulating a rollout with a learned dynamics model). Here, we consider two classes of exploration: *action-space noise*, by applying Gaussian noise to the actions of a deterministic policy, and *transition-space noise*, by eliciting risk-seeking behavior over modeling uncertainty in the simulated state transitions. Action-space noise can be applied during both the environment-interaction phase and the simulation phase, while the transition-space noise operates only in the simulation phase.

4.1 Exploration in Action Space for iLQR

Even though iLQR returns a deterministic policy, we can explore by inducing a probabilistic policy

$$\pi_t(\boldsymbol{a}_t | \boldsymbol{s}_t) \sim \mathcal{N}(\boldsymbol{a}_t^{\text{nom}} + \alpha \boldsymbol{k}_t + K_t(\boldsymbol{s}_t^{\text{nom}} - \boldsymbol{s}_t), \Sigma_{\pi_t}) \tag{1}$$

for some choice of time-dependent Gaussian noise covariance Σ_{π_t} Here, we consider two options for the covariance of the time-varying linear-Gaussian policies:

Action-fixed-covariance: We can set diag$(\Sigma_{\pi_t}) = [\epsilon_1^2, \cdots, \epsilon_{d_{\text{action}}}^2]$ for some fixed (i.e. not time-varying) $\epsilon_1^2, \dots, \epsilon_{d_{\text{action}}}^2$ where d_{action} is the dimensionality of the action space. This is a popular exploration method in model-free RL for continuous action spaces. In this work, we make the further assumption that the noise is isotropic, and set $\epsilon_i^2 = 0.2$ for $i = 1, \cdots, d_{\text{action}}$.

Action-maximum-entropy: A more principled choice for Σ_{π_t} for each timestep t is the curvature of the value function Q_t, i.e. $\Sigma_t = Q_{\boldsymbol{a}_t, \boldsymbol{a}_t}^{-1}$, which is shown in Levine & Koltun (2013) to be the solution of the maximum entropy control problem $\min \sum_{t=1}^{T} \mathbb{E}_{(\boldsymbol{s_t}, \boldsymbol{a_t}) \sim \pi(\boldsymbol{s_t}, \boldsymbol{a_t})}[c(\boldsymbol{s_t}, \boldsymbol{a_t}) - \mathcal{H}(\pi(\boldsymbol{a}_t | \boldsymbol{s}_t))]$ where $\mathcal{H}(\cdot)$ measures the entropy of action distribution induced by the policy $\pi_t(\boldsymbol{a}_t | \boldsymbol{s}_t)$. This solution has been used in subsequent work (e.g. Levine & Abbeel (2014); Levine et al. (2016)).

4.2 Exploration in transition space for iLQR

We can also explore through the transitions of the dynamics model by inducing a Gaussian distribution over our dynamics:

$$\mathbf{s}_{t+1} \sim \mathcal{N}(F(\boldsymbol{s}_t, \boldsymbol{a}_t), \Sigma_{\text{trans},t}) \tag{2}$$

We consider two options for $\Sigma_{\text{trans},t}$:

Transition-model-uncertainty: If the dynamics are modeled through a probabilistic model, a natural choice for $\Sigma_{\text{trans},t}$ is the *predictive uncertainty* of the learned probabilistic dynamics model, i.e. $\text{diag}(\Sigma_{\text{trans},t}) = \text{var}[\boldsymbol{s}_{t+1} \sim \hat{F}(\boldsymbol{s}_t, \boldsymbol{a}_t)]$. Intuitively, this encourages iLQR to explore states for which the learned model is uncertain.

Transition-fixed-covariance: Alternatively, we can induce exploration by fixing $\text{diag}(\Sigma_{\text{trans},t})$ to be some vector $[\epsilon_1^2, \dots, \epsilon_{d_{\text{state}}}^2]$, which is equivalent to assuming fixed isotropic uncertainty throughout the state space. In this work, we set $\epsilon_i^2 = 1.0$ for $i = 1, \dots, d_{\text{state}}$.

As described in Bechtle et al. (2019), in this setting, the variance of the trajectory cost $C(\pi) = \mathbb{E}_\pi[\sum_t^T c(\boldsymbol{s}_t, \boldsymbol{a}_t)]$ can be incorporated in iLQR by taking an exponential transform of the total cost function $C' = \exp[\sigma C(\pi)]$. A first order Taylor expansion of the cost gives

$$\frac{1}{\sigma}\log[C'] = \mathbb{E}[C] + \frac{\sigma}{2}\text{var}[C] + \dots \tag{3}$$

where $\text{var}[C]$ is the variance of the trajectory cost. Bechtle et al. (2019) solves this optimal control problem using a variant of the Riccati equation, following results from stochastic optimal control (Farshidian & Buchli (2015)).

The parameter σ weights the effect of $\Sigma_{\text{trans},t}$ in the Riccati equation. $\sigma = 0$ is equivalent to unmodified iLQR, while $\sigma < 0$ leads to exploratory behavior by incentivizing states which lead to trajectories with high uncertainty $\text{var}[C]$. In this work we set $\sigma = -0.3$, but extensively test the effect of modifying this parameter in Appendix A.

5 Experiments

For our experiments, we consider a PyBullet (Coumans & Bai (2016)) simulation of a 7 degree-of-freedom robotic manipulator, the Kuka iiwa7 robot arm. This physics simulator has been used in prior work to learn policies which transfer onto a real robot (Tan et al. (2018)). Our state representation includes the position and velocity of each joint, with dimensionality $d_{\text{state}} = 2 \cdot 7 = 14$, and the action is the applied joint torques after gravity compensation, with dimensionality $d_{\text{action}} = 7$. The cost function per timestep is the distance from the target joint state and a penalty on velocity and action norm:

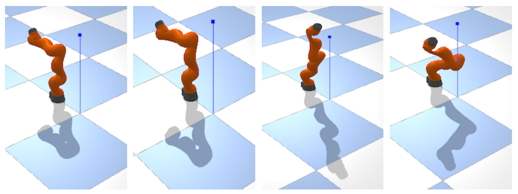
$$c(\mathbf{s}, \mathbf{a}) = q_{\text{pos}}||\mathbf{s}_{\text{pos}} - \mathbf{s}_{\text{target pos}}||_2^2 + q_{\text{vel}}||\mathbf{s}_{\text{vel}}||_2^2 + r_{\text{action}}||\mathbf{a}||_2^2 \tag{4}$$

with cost parameters $q_{\text{pos}} = 5$, $q_{\text{vel}} = 0.1$, and $r_{\text{action}} = 1\text{e}{-7}$. iLQR optimizes the summed total cost over the trajectory $\sum_{t=1}^T c(\boldsymbol{s}_t, \boldsymbol{a}_t)$.

The target joint state is generated using inverse kinematics for a random end-effector position. We generate targets three distances 0.1m, 0.45m, 0.8m away from the initial end-effector position as visualized by Figure 1, and 5 targets per distance, with a total of 15. These target end-effector positions signify increasingly difficult scenarios for the algorithm. We check to make sure all targets are achievable through inverse kinematics.

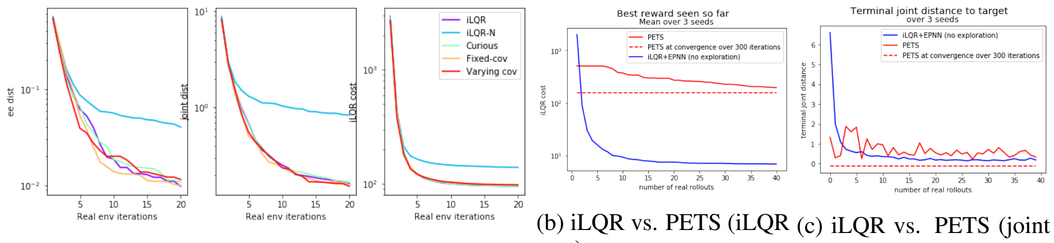
For each target, we initialize 10 random torque trajectories and run iLQR with learned EPNN

Figure 1: For each of 3 distances of $0.1, 0.45, 0.80$ meters representing 3 difficulty tiers, we generate 5 targets at that distance away from the initial end-effector position. We visualize the initial position and example target joint configurations for each.


(a) Initial (b) 0.1m (c) 0.45m (d) 0.80m

dynamics, measuring the quality of the solution through the iLQR cost over the real unrolled trajectory. To evaluate, we compute the mean and standard deviation of the trajectory cost across targets in each tier of difficulty.

5.1 Exploration for Model-Based RL



(a) iLQR with different exploration methods

(b) iLQR vs. PETS (iLQR cost)

(c) iLQR vs. PETS (joint dist)

Figure 2: MBRL learning curves with learned dynamics models. Here, we plot the best reward found so far averaged over 3 seeds. PETS takes much longer to reach convergence, so we additionally plot the best reward over any trajectory seen by PETS over the full 300 episode training.

We start out by analyzing the effect of adding exploration to the model-based RL loop (Algorithm 1). We note that action-space exploration can be applied both to the real rollout, which occurs in the *outer* loop as in line 6 of Algorithm 1, and within the simulated rollout, which occurs in the *inner* loop as in line 6 of Algorithm 3.

For completeness, we also provide a comparison to a state-of-the-art model-based RL algorithm, PETS, in Figure 2b tasked with optimizing the same cost function. PETS does receding-horizon MPC by replanning at every timestep, while iLQR optimizes over the full trajectory. As a result, iLQR trajectories can be unrolled on real systems, while PETS can take up to 1s for inferring actions at each time step. PETS quickly reaches a comparable solution to iLQR in terms of final distance to the desired joint position, but fails to reduce iLQR cost to comparable levels.

As a first step, we visualize the training curve for one of the randomly-generated targets in Figure 2a, to give some insights into convergence behavior for different exploration behaviors. Based on these plots, the first result is that, at least for this target, there is little to no benefit of adding exploration to the MBRL loop.

As a second step we perform all possible combinations of the two types of exploration methods and show results in Table 1.

Table 1: Exploration with EPNN model.
Full trajectory cost, averaged over targets with equal distance from initial end-effector

Inner	Outer	0.10m	std	0.45m	std	0.80m	std
No exploration		**16.77**	8.05	**190.12**	139.45	**383.11**	53.53
Trans-model-uncertainty		18.18	8.03	205.18	146.09	391.96	43.59
Trans-fixed-cov		18.70	9.22	196.26	138.44	397.74	53.47
Action-fixed-cov		38.89	9.40	230.08	133.09	456.40	64.71
Action-max-ent		236.98	25.99	471.01	132.17	755.74	63.21
	Action-fixed-cov	20.76	8.34	205.85	152.07	395.90	49.62
	Action-max-ent	112.96	11.34	284.14	138.00	481.44	42.25
Trans-model-uncertainty	Action-fixed-cov	20.63	8.05	194.86	135.92	393.06	41.09
Trans-fixed-cov	Action-fixed-cov	21.65	8.06	231.65	148.29	406.17	40.65

The most significant findings across this large-scale experiment:

- Both action-phase exploration methods, produce better results when applied in the real-system rollout phase (outer loop). Action-fixed-cov performs better than Action-max-ent.

- All transition-exploration based methods perform better than the action-based exploration methods.
- However, most notably, *none* of the exploration methods improve the performance of our basic MBRL loop without exploration.

5.2 Exploration for Trajectory Optimization Using Ground-Truth Dynamics

In an attempt to get a better understanding of the surprising result that exploration does not aid MBRL, we aim to isolate the effect of policy optimization from dynamics model learning. In our next set of experiments, we use a *ground-truth* dynamics model, wherein the simulator itself is used as the dynamics model. We use the finite-difference approximation of the model gradients when required by iLQR. We run the same set of experiments as above, but only with inner-loop exploration. Results are summarized in Table 2.

Table 2: Exploration with ground-truth model.
Full trajectory cost, averaged over targets with equal distance from initial end-effector

Algorithm	0.10m	std	0.45m	std	0.80m	std
None	36.71	60.75	205.63	128.92	**512.73**	210.29
Action-fixed-cov	73.70	60.76	222.40	105.10	563.15	218.87
Action-max-ent	214.53	79.96	432.82	90.46	801.94	186.32
Trans-fixed-cov	**24.52**	25.67	**196.81**	109.81	535.25	190.15

Here, we clearly see a benefit of adding transition-exploration for the first set of targets. Second, we note the high variance of iLQR without exploration results, clearly demonstrating the inability of iLQR to find good solutions consistently, across all target difficulties. iLQR with fixed-covariance and transition-space exploration leads to better mean results, as well as reduced variance of the solutions for nearby targets.

This exploration bonus likely only comes up because of the local nature of our optimizer and our choice of globalization strategy (simple line search). This effect might be mitigated when using more advanced (and more complex) globalization strategies (Nocedal & Wright (2006)) or using a multiple shooting version of iLQR, less sensitive to initial conditions (Mastalli et al. (2019)). Nevertheless, most optimizers in high-dimensional spaces are necessarily local when used on real robots for computational efficiency reasons and will remain prone to getting stuck in local minima even when using sampling methods.

While this set of experiments confirms our intuition that exploration can help, it is unclear why this effect is not visible during the MBRL experiments. However, when taking a step back, and looking at all results generated so far, Tables 1 and 2, we observe a surprising result: the MBRL loop via regular iLQR, which learns a dynamics model via EPNN, actually outperforms iLQR on ground truth dynamics. We discuss this below.

5.3 Exploration through exploitation for model-based RL

For clarity, we present the key results again, but from a different perspective. We compare iLQR on ground truth dynamics with and without exploration with iLQR on learned dynamics (Table 3).

Table 3: iLQR with learned dynamics vs ground truth dynamics.
Full trajectory cost, averaged over targets with equal distance from initial end-effector

Problem setting	0.10m	std	0.45m	std	0.80m	std
Ground-truth with no exploration	36.71	60.75	205.63	128.92	512.73	210.29
Ground-truth with fixed-cov	24.52	25.67	196.81	109.81	535.25	190.15
EPNN with no exploration	**16.77**	8.05	**190.12**	139.45	**383.11**	53.53

These results show that using iLQR within a MBRL loop to learn the dynamics model leads to a better performance consistently.

This effect can be explained as follows: starting from a randomly initialized dynamics model in MBRL has an unexpected benefit: it results in an implicit exploration. During policy optimization, the poor predictions from the dynamics model will initially lead to undirected policies, that perform poorly on the task but creates diverse data to update the dynamics model, and a better initialization for the optimizer. Because of this implicit exploration from the bad dynamics model, the advantage of explicit exploration is less visible in MBRL. Incorporating fixed transition-covariance in the trajectory optimization with ground-truth dynamics improves its performance, bringing it closer to learned dynamics. This reinforces our hypothesis that the poor performance on ground-truth is due to lack of exploration.

5.3.1 Model Generalization to New Targets

Finally, we evaluate the generalization capabilities of the dynamics models learned from the various explorations methods. For each dynamics model learned for targets of distance d meters, we use the standard iLQR algorithm (with no exploratory behavior) to optimize a policy for *other* targets at the same distance d. We roll out the policy onto the real environment once (i.e. no model learning). In this way, we aim to measure the ability of the dynamics models themselves to generalize without additional training.

Table 4: EPNN model generalization. Models are taken after training and tested on new targets.
Full trajectory cost, averaged over targets with equal distance from initial end-effector

Algorithm for dynamics data	0.10m	std	0.45m	std	0.80m	std
EPNN without exploration	56.21	20.49	365.33	218.38	1402.28	1246.29
Action-fixed-cov (inner)	**54.93**	16.62	**307.65**	145.74	1414.48	1556.82
Trans-model-uncertainty	58.78	18.07	329.90	150.20	1539.96	1547.73
Trans-fixed-cov	59.18	24.09	325.34	152.16	**1267.18**	1011.21

The results show that exploration does provide some benefit - it helps learn somewhat better models, although there is no consistent “winner”. Also, note that the trained models do not actually generalize well to new targets, as compared to EPNN trained on a target (old cost for $d = 0.10$ was 16.77 ± 8.05), or using ground truth (cost for $d = 0.10$ was 36.71 ± 60.75).

Thus, we conclude that though iLQR with EPNN can perform better than iLQR with ground-truth *for a particular target*, the learned EPNN models fail to generalize to new targets without retraining. This demonstrates that the models learned in our model-based RL loop are effective *local* models: highly specific to the task for which it is trained but not well-suited for solving new tasks without further training, in contrast to global models like the ground-truth dynamics.

6 Conclusion

In this work, we studied the effect of exploration in different stages of a model-based RL loop. We added action-space noise when executing a policy on the real system, as well as when rolling out on the simulated dynamics. Whether isotropic Gaussian noise or the maximum-entropy solution, action-space noise applied to the optimized policy degrades performance across all our experiments. We also added transition-space noise to the dynamics when doing trajectory optimization, using a risk-seeking iLQR formulation from literature. In this case, we observe that iLQR with learned dynamics and no exploration performs better than all the transition-space exploration techniques considered. In fact, for our experimental setup, using ground-truth dynamics without exploration performs worse than EPNN with iLQR due to the exploratory nature of imperfect dynamics models.

Although our conclusions are limited by the choice of policy optimization algorithms, model representations and experimental environments, our results shed lights on effects that are seldom discussed in MBRL. Future work will explore the effect of exploration with different classes of MBRL algorithms and other robotic tasks.

REFERENCES

Sarah Bechtle, Akshara Rai, Yixin Lin, Ludovic Righetti, and Franziska Meier. Curious ilqr: Resolving uncertainty in model-based rl. *arXiv preprint arXiv:1904.06786*, 2019.

Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pp. 679–684, 1957.

Joschka Boedecker, Jost Tobias Springenberg, Jan Wülfing, and Martin Riedmiller. Approximate real-time optimal control based on sparse gaussian process models. In *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pp. 1–8. IEEE, 2014.

Miroslav Bogdanovic and Ludovic Righetti. Learning to explore in motion and interaction tasks. *arXiv preprint arXiv:1908.03731*, 2019.

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018a.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018b.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pp. 4754–4765, 2018.

Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth Stanley, and Jeff Clune. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Advances in Neural Information Processing Systems*, pp. 5027–5038, 2018.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. *GitHub repository*, 2016.

Farbod Farshidian and Jonas Buchli. Risk sensitive, nonlinear optimal control: Iterative linear exponential-quadratic optimal control with gaussian noise. *arXiv preprint arXiv:1512.07173*, 2015.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1352–1361. JMLR. org, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Todd Hester and Peter Stone. Intrinsically motivated model learning for developing curious robots. *Artificial Intelligence*, 247:170–186, 2017.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pp. 6402–6413, 2017.

Sergey Levine and Pieter Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems*, pp. 1071–1079, 2014.

Sergey Levine and Vladlen Koltun. Guided policy search. In *International Conference on Machine Learning*, pp. 1–9, 2013.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO (1)*, pp. 222–229, 2004.

Manuel Lopes, Tobias Lang, Marc Toussaint, and Pierre-Yves Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. In *Advances in Neural Information Processing Systems*, pp. 206–214, 2012.

Carlos Mastalli, Rohan Budhiraja, Wolfgang Merkt, Guilhem Saurel, Bilal Hammoud, Maximilien Naveau, Justin Carpentier, Sethu Vijayakumar, and Nicolas Mansard. Crocoddyl: An efficient and versatile framework for multi-contact optimal control. *arXiv preprint arXiv:1909.04947*, 2019.

Teodor Mihai Moldovan, Sergey Levine, Michael I Jordan, and Pieter Abbeel. Optimism-driven exploration for nonlinear systems. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3239–3246. IEEE, 2015.

Michael Neunert, Cedric De Crousaz, Fadri Furrer, Mina Kamel, Farbod Farshidian, Roland Siegwart, and Jonas Buchli. Fast nonlinear model predictive control for unified trajectory optimization and tracking. In *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 1398–1404. IEEE, 2016.

Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 16–17, 2017.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. *arXiv preprint arXiv:1906.04161*, 2019.

Martin Pelikan, David E Goldberg, and Erick Cantú-Paz. Boa: The bayesian optimization algorithm. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*, pp. 525–532. Morgan Kaufmann Publishers Inc., 1999.

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

Thomas Rückstieß, Martin Felder, and Jürgen Schmidhuber. State-dependent exploration for policy gradient methods. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 234–249. Springer, 2008.

Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pp. 222–227, 1991.

Jürgen Schmidhuber. Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. *Connection Science*, 18(2):173–187, 2006.

Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. Model-based active exploration. *arXiv preprint arXiv:1810.12162*, 2018.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pp. 2951–2959, 2012.

Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 2. MIT press Cambridge, 1998.

Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

Julian Viereck, Jules Kozolinsky, Alexander Herzog, and Ludovic Righetti. Learning a structured neural network policy for a hopping task. *IEEE Robotics and Automation Letters*, 3(4):4092–4099, 2018.

Marco Wiering and Jürgen Schmidhuber. Efficient model-based exploration. In *Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, volume 6, pp. 223–228, 1998.

Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. PhD thesis, figshare, 2010.

A INFLUENCE OF RISK-SEEKING PARAMETER

We measure the effect of modifying σ to produce varying levels of risk-seeking behavior, on ground truth and EPNN dynamics. In the case of ground-truth, increased exploration helps achieve better performance, in line with our previous observations. However, for EPNN all values of σ perform equally if not worse than $\sigma = 0$.

A.1 TRAJECTORY OPTIMIZATION ON GROUND-TRUTH DYNAMICS

Table 5: Varying σ (ground-truth model)
Full trajectory cost, averaged over targets with equal distance from initial end-effector

Algorithm	0.10m	std	0.45m	std	0.80m	std
σ=0.0	36.71	60.75	205.63	128.92	**512.73**	210.29
Trans-fixed-cov σ=-0.3	24.52	25.67	196.81	109.81	535.25	190.15
Trans-fixed-cov σ=-0.5	**21.67**	11.09	**186.83**	94.14	548.62	207.91

A.2 MBRL LOOP

Table 6: Varying σ (EPNN)
Full trajectory cost, averaged over targets with equal distance from initial end-effector

Algorithm	0.10m	std	0.45m	std	0.80m	std
σ=0.0	**16.77**	8.05	**190.12**	139.45	**383.11**	53.53
Trans-fixed-cov σ=-0.3	18.70	9.22	196.26	138.44	397.74	53.47
Trans-fixed-cov σ=-0.5	18.47	8.26	199.48	140.16	404.73	40.94