arXiv:1805.12114v2 [cs.LG] 2 Nov 2018

Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models

Kurtland Chua **Roberto Calandra** **Rowan McAllister** **Sergey Levine**
Berkeley Artificial Intelligence Research
University of California, Berkeley
{kchua, roberto.calandra, rmcallister, svlevine}@berkeley.edu

Abstract

Model-based reinforcement learning (RL) algorithms can attain excellent sample efficiency, but often lag behind the best model-free algorithms in terms of asymptotic performance. This is especially true with high-capacity parametric function approximators, such as deep networks. In this paper, we study how to bridge this gap, by employing uncertainty-aware dynamics models. We propose a new algorithm called probabilistic ensembles with trajectory sampling (PETS) that combines uncertainty-aware deep network dynamics models with sampling-based uncertainty propagation. Our comparison to state-of-the-art model-based and model-free deep RL algorithms shows that our approach matches the asymptotic performance of model-free algorithms on several challenging benchmark tasks, while requiring significantly fewer samples (e.g., 8 and 125 times fewer samples than Soft Actor Critic and Proximal Policy Optimization respectively on the half-cheetah task).

1 Introduction

Reinforcement learning (RL) algorithms provide for an automated framework for decision making and control: by specifying a high-level objective function, an RL algorithm can, in principle, automatically learn a control policy that satisfies this objective. This has the potential to automate a range of applications, such as autonomous vehicles and interactive conversational agents. However, current model-free reinforcement learning algorithms are quite expensive to train, which often limits their application to simulated domains [Mnih et al., 2015, Lillicrap et al., 2016, Schulman et al., 2017], with a few exceptions [Kober and Peters, 2009, Levine et al., 2016]. A promising direction for reducing sample complexity is to explore model-based reinforcement learning (MBRL) methods, which proceed by first acquiring a predictive model of the world, and then using that model to make decisions [Atkeson and Santamaría, 1997, Kocijan et al., 2004, Deisenroth et al., 2014]. MBRL is appealing because the dynamics model is reward-independent and therefore can generalize to new tasks in the same environment, and it can easily benefit from all of the advances in deep supervised learning to utilize high-capacity models. However, the asymptotic performance of MBRL methods on common benchmark tasks generally lags behind model-free methods. That is, although MBRL methods tend to learn more quickly, they also tend to converge to less optimal solutions.

In this paper, we take a step toward narrowing the gap between model-based and model-free RL methods. Our approach is based on several observations that, though relatively simple, are critical for good performance. We first observe that model capacity is a critical ingredient in the success of MBRL methods: while efficient models such as Gaussian processes can learn extremely quickly, they struggle to represent very complex and discontinuous dynamical systems [Calandra et al., 2016]. By contrast, neural network (NN) models can scale to large datasets with high-dimensional inputs, and can represent such systems more effectively. However, NNs struggle with the opposite problem:

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

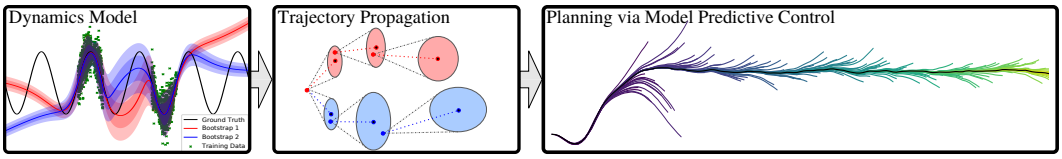


Figure 1: Our method (PE-TS): **Model**: Our probabilistic ensemble (PE) dynamics model is shown as an ensemble of two bootstraps (bootstrap disagreement far from data captures epistemic uncertainty: our subjective uncertainty due to a lack of data), each a probabilistic neural network that captures aleatoric uncertainty (inherent variance of the observed data). **Propagation**: Our trajectory sampling (TS) propagation technique uses our dynamics model to re-sample each particle (with associated bootstrap) according to its probabilistic prediction at each point in time, up until horizon T. **Planning**: At each time step, our MPC algorithm computes an optimal action sequence, applies the first action in the sequence, and repeats until the task-horizon.

to learn fast means to learn with few data and NNs tend to overfit on small datasets, making poor predictions far into the future. For this reason, MBRL with NNs has proven exceptionally challenging.

Our second observation is that this issue can, to a large extent, be mitigated by properly incorporating uncertainty into the dynamics model. While a number of prior works have explored uncertainty-aware deep neural network models [Neal, 1995, Lakshminarayanan et al., 2017], including in the context of RL [Gal et al., 2016, Depeweg et al., 2016], our work is, to our knowledge, the first to bring these components together in a deep MBRL framework that reaches the asymptotic performance of state-of-the-art model-free RL methods on benchmark control tasks.

Our main contribution is an MBRL algorithm called probabilistic ensembles with trajectory sampling (PETS)[1] summarized in Figure 1 with high-capacity NN models that incorporate uncertainty via an ensemble of bootstrapped models, where each model encodes *distributions* (as opposed to point predictions), rivaling the performance of model-free methods on standard benchmark control tasks at a fraction of the sample complexity. An advantage of PETS over prior probabilistic MBRL algorithms is an ability to isolate two distinct classes of uncertainty: aleatoric (inherent system stochasticity) and epistemic (subjective uncertainty, due to limited data). Isolating epistemic uncertainty is especially useful for directing exploration [Thrun, 1992], although we leave this for future work. Finally, we present a systematic analysis of how incorporating uncertainty into MBRL with NNs affects performance, during both model training and planning. We show, that PETS' particular treatment of uncertainty significantly reduces the amount of data required to learn a task, e.g., eight times fewer data on half-cheetah compared to the model-free Soft Actor Critic algorithm [Haarnoja et al., 2018].

2 Related work

Model choice in MBRL is delicate: we desire effective learning in both low-data regimes (at the beginning) and high-data regimes (in the later stages of the learning process). For this reason, Bayesian nonparametric models, such as Gaussian processes (GPs), are often the model of choice in MBRL, especially in low-dimensional problems where data efficiency is critical [Kocijan et al., 2004, Ko et al., 2007, Nguyen-Tuong et al., 2008, Grancharova et al., 2008, Deisenroth et al., 2014, Kamthe and Deisenroth, 2018]. However, such models introduce additional assumptions on the system, such as the smoothness assumption inherent in GPs with squared-exponential kernels [Rasmussen and Kuss, 2003]. Parametric function approximators have also been used extensively in MBRL [Hernandaz and Arkun, 1990, Miller et al., 1990, Lin, 1992, Draeger et al., 1995], but were largely supplanted by Bayesian models in recent years. Methods based on local models, such as guided policy search algorithms [Levine et al., 2016, Finn et al., 2016, Chebotar et al., 2017], can efficiently train NN policies, but use time-varying linear models, which only locally model the system dynamics. Recent improvements in parametric function approximators, such as NNs, suggest that such methods are worth revisiting [Baranes and Oudeyer, 2013, Fu et al., 2016, Punjani and Abbeel, 2015, Lenz et al., 2015, Agrawal et al., 2016, Gal et al., 2016, Depeweg et al., 2016, Williams et al., 2017, Nagabandi et al., 2017]. Unlike Gaussian processes, NNs have constant-time inference and tractable training in the large data regime, and have the potential to represent more complex functions, including non-

[1] Code available `https://github.com/kchua/handful-of-trials`

smooth dynamics that are often present in robotics [Fu et al., 2016, Mordatch et al., 2016, Nagabandi et al., 2017]. However, most works that use NNs focus on deterministic models, consequently suffering from overfitting in the early stages of learning. For this reason, our approach is able to achieve even higher data-efficiency than prior deterministic MBRL methods such as Nagabandi et al. [2017].

Constructing good Bayesian NN models remains an open problem [MacKay, 1992, Neal, 1995, Osband, 2016, Guo et al., 2017], although recent promising work exists on incorporating dropout [Gal et al., 2017], ensembles [Osband et al., 2016, Lakshminarayanan et al., 2017], and α-divergence [Hernández-Lobato et al., 2016]. Such probabilistic NNs have previously been used for control, including using dropout Gal et al. [2016], Higuera et al. [2018] and α-divergence Depeweg et al. [2016]. In contrast to these prior methods, our experiments focus on more complex tasks with challenging dynamics – including contact discontinuities – and we compare directly to prior model-based and model-free methods on standard benchmark problems, where our method exhibits asymptotic performance that is comparable to model-free approaches.

3 Model-based reinforcement learning

We now detail the MBRL framework and the notation used. Adhering to the Markov decision process formulation [Bellman, 1957], we denote the state $\boldsymbol{s} \in \mathbb{R}^{d_s}$ and the actions $\boldsymbol{a} \in \mathbb{R}^{d_a}$ of the system, the reward function $r(\boldsymbol{s}, \boldsymbol{a})$, and we consider the dynamic systems governed by the transition function $f_{\boldsymbol{\theta}} : \mathbb{R}^{d_s+d_a} \mapsto \mathbb{R}^{d_s}$ such that given the current state $\boldsymbol{s}_t$ and current input $\boldsymbol{a}_t$, the next state $\boldsymbol{s}_{t+1}$ is given by $\boldsymbol{s}_{t+1} = f(\boldsymbol{s}_t, \boldsymbol{a}_t)$. For probabilistic dynamics, we represent the conditional distribution of the next state given the current state and action as some parameterized distribution family: $f_{\boldsymbol{\theta}}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) = \Pr(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\theta})$, overloading notation. Learning forward dynamics is thus the task of fitting an approximation $\widetilde{f}$ of the true transition function f, given the measurements $\mathcal{D} = \{(\boldsymbol{s}_n, \boldsymbol{a}_n), \boldsymbol{s}_{n+1}\}_{n=1}^{N}$ from the real system.

Once a dynamics model $\widetilde{f}$ is learned, we use $\widetilde{f}$ to predict the distribution over state-trajectories resulting from applying a sequence of actions. By computing the expected reward over state-trajectories, we can evaluate multiple candidate action sequences, and select the optimal action sequence to use. In Section 4 we discuss multiple methods for modeling the dynamics, and in Section 5 we detail how to compute the distribution over state-trajectories given a candidate action sequence.

4 Uncertainty-aware neural network dynamics models

This section describes several ways to model the task's true (but unknown) dynamic function, including our method: an ensemble of bootstrapped probabilistic neural networks. Whilst uncertainty-aware dynamics models have been explored in a number of prior works [Gal et al., 2016, Depeweg et al., 2016], the particular details of the implementation and design decisions in regard incorporation of uncertainty have not been rigorously analyzed empirically. As a result, prior work has generally found that expressive parametric models, such as deep neural networks, generally do not produce model-based RL algorithms that are competitive with their model-free counterparts in terms of asymptotic performance [Nagabandi et al., 2017], and often even found that simpler time-varying linear models can outperform expressive neural network models [Levine et al., 2016, Gu et al., 2016].

Table 1: Model uncertainties captured.

Model	**Aleatoric uncertainty**	**Epistemic uncertainty**
Baseline Models		
Deterministic NN (D)	No	No
Probabilistic NN (P)	Yes	No
Deterministic ensemble NN (DE)	No	Yes
Gaussian process baseline (GP)	Homoscedastic	Yes
Our Model		
Probabilistic ensemble NN (PE)	**Yes**	**Yes**

Any MBRL algorithm must select a class of model to predict the dynamics. This choice is often crucial for an MBRL algorithm, as even small bias can significantly influence the quality of the corresponding controller [Atkeson and Santamaría, 1997, Abbeel et al., 2006]. A major challenge is building a model that performs well in low and high data regimes: in the early stages of training, data is scarce, and highly expressive function approximators are liable to overfit; In the later stages of training, data is plentiful, but for systems with complex dynamics, simple function approximators might underfit. While Bayesian models such as GPs perform well in low-data regimes, they do not scale favorably

with dimensionality and often use kernels ill-suited for discontinuous dynamics [Calandra et al., 2016], which is typical of robots interacting through contacts.

In this paper, we study how expressive NNs can be incorporated into MBRL. To account for uncertainty, we study NNs that model two types of uncertainty. The first type, aleatoric uncertainty, arises from *inherent stochasticities* of a system, e.g. observation noise and process noise. Aleatoric uncertainty can be captured by outputting the parameters of a parameterized distribution, while still training the network discriminatively. The second type – epistemic uncertainty – corresponds to *subjective uncertainty* about the dynamics function, due to a lack of sufficient data to uniquely determine the underlying system exactly. In the limit of infinite data, epistemic uncertainty should vanish, but for datasets of finite size, subjective uncertainty remains when predicting transitions. It is precisely the subjective epistemic uncertainty which Bayesian modeling excels at, which helps mitigate overfitting. Below, we describe how we use combinations of 'probabilistic networks' to capture aleatoric uncertainty and 'ensembles' to capture epistemic uncertainty. Each combination is summarized in Table 1.

Probabilistic neural networks (P) We define a *probabilistic* NN as a network whose output neurons simply parameterize a probability distribution function, capturing aleatoric uncertainty, and should not be confused with Bayesian inference. We use the negative log prediction probability as our loss function $\text{loss}_{\text{P}}(\boldsymbol{\theta}) = -\sum_{n=1}^{N} \log \widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{s}_{n+1}|\boldsymbol{s}_n, \boldsymbol{a}_n)$. For example, we might define our predictive model to output a Gaussian distribution with diagonal covariances parameterized by $\boldsymbol{\theta}$ and conditioned on $\boldsymbol{s}_n$ and $\boldsymbol{a}_n$, i.e.: $\widetilde{f} = \Pr(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) = \mathcal{N}(\boldsymbol{\mu}_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t), \boldsymbol{\Sigma}_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t))$. Then the loss becomes

$$\text{loss}_{\text{Gauss}}(\boldsymbol{\theta}) = \sum_{n=1}^{N} [\mu_{\boldsymbol{\theta}}(\boldsymbol{s}_n, \boldsymbol{a}_n) - \boldsymbol{s}_{n+1}]^\top \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{s}_n, \boldsymbol{a}_n)[\mu_{\boldsymbol{\theta}}(\boldsymbol{s}_n, \boldsymbol{a}_n) - \boldsymbol{s}_{n+1}] + \log\det \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{s}_n, \boldsymbol{a}_n). \quad (1)$$

Such network outputs, which in our particular case parameterizes a Gaussian distribution, models aleatoric uncertainty, otherwise known as heteroscedastic noise (meaning the output distribution is a function of the input). However, it does not model epistemic uncertainty, which cannot be captured with purely discriminative training. Choosing a Gaussian distribution is a common choice for continuous-valued states, and reasonable if we assume that any stochasticity in the system is unimodal. However, in general, any tractable distribution class can be used. To provide for an expressive dynamics model, we can represent the parameters of this distribution (e.g., the mean and covariance of a Gaussian) as nonlinear, parametric functions of the current state and action, which can be arbitrarily complex but deterministic. This makes it feasible to incorporate NNs into a probabilistic dynamics model even for high-dimensional and continuous states and actions. Finally, an under-appreciated detail of probabilistic networks is that their variance has *arbitrary* values for out-of-distribution inputs, which can disrupt planning. We discuss how to mitigate this issue in Appendix A.1.

Deterministic neural networks (D) For comparison, we define a deterministic NN as a special-case probabilistic network that outputs delta distributions centered around point predictions denoted as $\widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t)$: $\widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) = \Pr(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) = \delta(\boldsymbol{s}_{t+1} = \widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t))$, trained using the MSE loss: $\text{loss}_D(\boldsymbol{\theta}) = \sum_{n=1}^{N} \|\boldsymbol{s}_{n+1} - \widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{s}_n, \boldsymbol{a}_n)\|$. Although MSE can be interpreted as $\text{loss}_{\text{P}}(\boldsymbol{\theta})$ with a Gaussian model of fixed unit variance, in practice this variance cannot be used for uncertainty-aware propagation, since it does not correspond to any notion of uncertainty (e.g., a deterministic model with infinite data would be adding variance to particles for no good reason).

Ensembles (DE and PE) A principled means to capture epistemic uncertainty is with Bayesian inference. Whilst accurate Bayesian NN inference is possible with sufficient compute [Neal, 1995], approximate inference methods [Blundell et al., 2015, Gal et al., 2017, Hernández-Lobato and Adams, 2015] have enjoyed recent popularity given their simpler implementation and faster training times. Ensembles of bootstrapped models are even simpler still: given a base model, no additional (hyper-)parameters need be tuned, whilst still providing reasonable uncertainty estimates [Efron and Tibshirani, 1994, Osband, 2016, Kurutach et al., 2018]. We consider ensembles of B-many bootstrap models, using $\boldsymbol{\theta}_b$ to refer to the parameters of our b^{th} model $\widetilde{f}_{\boldsymbol{\theta}_b}$. Ensembles can be composed of deterministic models (DE) or probabilistic models (PE) – as done by Lakshminarayanan et al. [2017] – both of which define predictive probability distributions: $\widetilde{f}_{\boldsymbol{\theta}} = \frac{1}{B}\sum_{b=1}^{B} \widetilde{f}_{\boldsymbol{\theta}_b}$. A visual example is provided in Appendix A.2. Each of our bootstrap models have their unique dataset $\mathbb{D}_b$, generated by

sampling (with replacement) N times the dynamics dataset recorded so far $\mathbb{D}$, where N is the size of $\mathbb{D}$. We found $B = 5$ sufficient for all our experiments. To validate the number of layers and neurons of our models, we can visualize one-step predictions (e.g. Appendix A.3).

5 Planning and control with learned dynamics

This section describes different ways uncertainty can be incorporated into planning using probabilistic dynamics models. Once a model $\widetilde{f}_{\boldsymbol{\theta}}$ is learned, we can use it for control by predicting the future outcomes of candidate policies or actions and then selecting the particular candidate that is predicted to result in the highest reward. MBRL planning in discrete time over long time horizons is generally performed by using the dynamics model to recursively predict how an estimated Markov state will evolve from one time step to the next, e.g.: $\boldsymbol{s}_{t+2} \sim \Pr(\boldsymbol{s}_{t+2}|\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1})$ where $\boldsymbol{s}_{t+1} \sim \Pr(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$. When planning, we might consider each action $\boldsymbol{a}_t$ to be a function of state, forming a policy $\pi : \boldsymbol{s}_t \rightarrow \boldsymbol{a}_t$, a function to optimize. Alternatively, we can plan and optimize for a sequence of actions, a process called model predictive control (MPC) [Camacho and Alba, 2013]. We use MPC in our own experiments for several reasons, including implementation simplicity, lower computational burden (no gradients), and no requirement to specify the task-horizon in advance, whilst achieving the same data-efficiency as Gal et al. [2016] who used a Bayesian NN with a policy to learn the cart-pole task in 2000 time steps. Our full algorithm is summarized in Section 6.

Given the state of the system $\boldsymbol{s}_t$ at time t, the prediction horizon T of the MPC controller, and an action sequence $\boldsymbol{a}_{t:t+T} \doteq \{\boldsymbol{a}_t, \ldots, \boldsymbol{a}_{t+T}\}$; the *probabilistic* dynamics model $\widetilde{f}$ induces a distribution over the resulting trajectories $\boldsymbol{s}_{t:t+T}$. At each time step t, the MPC controller applies the first action $\boldsymbol{a}_t$ of the sequence of optimized actions $\arg\max_{\boldsymbol{a}_{t:t+T}} \sum_{\tau=t}^{t+T} \mathbb{E}_{\widetilde{f}}[r(\boldsymbol{s}_\tau, \boldsymbol{a}_\tau)]$. A common technique to compute the optimal action sequence is a random sampling shooting method, due to its parallelizability and ease of implementation. Nagabandi et al. [2017] use deterministic NN models and MPC with random shooting to achieve data efficient control in higher dimensional tasks than what is feasible for GPs to model. Our work improves upon Nagabandi et al. [2017]'s data efficiency in two ways: First, we capture uncertainty in modeling and planning, to prevent overfitting in the low-data regime. Second, we use CEM [Botev et al., 2013] instead of random-shooting, which samples actions from a distribution closer to previous action samples that yielded high reward.

Computing the expected trajectory reward using recursive state prediction in closed-form is generally intractable. Multiple approaches to approximate uncertainty propagation can be found in the literature [Girard et al., 2002, Quiñonero-Candela et al., 2003]. These approaches can be categorized by how they represent the state distribution: deterministic, particle, and parametric methods. Deterministic methods use the mean prediction and ignore the uncertainty, particle methods propagate a set of Monte Carlo samples, and parametric methods include Gaussian or Gaussian mixture models, etc. Although parametric distributions have been successfully used in MBRL [Deisenroth et al., 2014], experimental results [Kupcsik et al., 2013] suggest that particle approaches can be competitive both computationally and in terms of accuracy, without making strong assumptions about the distribution used. Hence, we use particle-based propagation, specifically suited to our PE dynamics model which distinguishes two types of uncertainty, detailed in Section 5.1. Unfortunately, little prior work has empirically compared the design decisions involved in choosing the particular propagation method. Thus, we compare against several baselines in Section 5.2. Visual examples are provided in Appendix A.4.

5.1 Our state propagation method: trajectory sampling (TS)

Our method to predict plausible state trajectories begins by creating P particles from the current state, $\boldsymbol{s}^p_{t=0} = \boldsymbol{s}_0 \, \forall \, p$. Each particle is then propagated by: $\boldsymbol{s}^p_{t+1} \sim \widetilde{f}_{\boldsymbol{\theta}_{b(p,t)}}(\boldsymbol{s}^p_t, \boldsymbol{a}_t)$, according to a particular bootstrap $b(p,t)$ $in\{1, \ldots, B\}$, where B is the number of bootstrap models in the ensemble. A particle's bootstrap index can potentially change as a function of time t. We consider two TS variants:

- **TS1** refers to particles uniformly re-sampling a bootstrap per time step. If we were to consider an ensemble as a Bayesian model, the particles would be effectively continually re-sampling from the approximate *marginal posterior* of plausible dynamics. We consider TS1's bootstrap re-sampling to place a soft restriction on trajectory multimodality: particles separation cannot be attributed to the *compounding* effects of differing bootstraps using TS1.

- **TS∞** refers to particle bootstraps never changing during a trial. An ensemble is a collection of plausible models, which together represent the *subjective* uncertainty in function space of the true dynamics function f, which we assume is time invariant. TS∞ captures such time invariance since each particle's bootstrap index is made consistent over time. An advantage of using TS∞ is that aleatoric and epistemic uncertainties are separable [Depeweg et al., 2018]. Specifically, aleatoric state variance is the average variance of particles of same bootstrap, whilst epistemic state variance is the variance of the average of particles of same bootstrap indexes. Epistemic is the 'learnable' type of uncertainty, useful for directed exploration [Thrun, 1992]. Without a way to distinguish epistemic uncertainty from aleatoric, an exploration algorithm (e.g. Bayesian optimization) might mistakingly choose actions with high predicted reward-variance 'hoping to learn something' when in fact such variance is caused by persistent and irreducible system stochasticity offering zero exploration value.

Both TS variants can capture multi-modal distributions and can be used with any probabilistic model. We found $P = 20$ and $B = 5$ sufficient in all our experiments.

5.2 Baseline state propagation methods for comparison

To validate our state propagation method, in the experiments of Section 7.2 we compare against four alternative state propagation methods, which we now discuss.

Expectation (E) To judge the importance of our TS method using multiple particles to represent a distribution we compare against the aforementioned deterministic propagation technique. The simplest way to plan is iteratively propagating the expected prediction at each time step (ignoring uncertainty) $\boldsymbol{s}_{t+1} = \mathbb{E}[\widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t)]$. An advantage of this approach over TS is reduced computation and simple implementation: only a single particle is propagated. The main disadvantage of choosing E over TS is that small model biases can compound quickly over time, with no way to tell the quality of the state estimate.

Moment matching (MM) Whilst TS's particles can represent multimodal distributions, forcing a unimodal distribution via moment matching (MM) can (in some cases) benefit MBRL data efficiency [Gal et al., 2016]. Although unclear why, Gal et al. [2016] (who use Gaussian MM) hypothesize this effect may be caused by smoothing of the loss surface and implicitly penalizing multi-modal distributions (which often only occur with uncontrolled systems). To test this hypothesis we use Gaussian MM as a baseline and assume independence between bootstraps and particles for simplicity $\boldsymbol{s}^p_{t+1} \overset{iid}{\sim} \mathcal{N}\left(\mathbb{E}_{p,b}\left[\boldsymbol{s}^{p,b}_{t+1}\right], \mathbb{V}_{p,b}\left[\boldsymbol{s}^{p,b}_{t+1}\right]\right)$, where $\boldsymbol{s}^{p,b}_{t+1} \sim \widetilde{f}_{\boldsymbol{\theta}_b}(\boldsymbol{s}^p_t, \boldsymbol{a}_t)$. Future work might consider other distributions too, such as the Laplace distribution.

Distribution sampling (DS) The previous MM approach made a strong unimodal assumption about state distributions: the state distribution at each time step was re-cast to Gaussian. A softer restriction on multimodality – between MM and TS – is to moment match w.r.t. the bootstraps only (noting the particles are otherwise independent if $B = 1$). This means that we effectively smooth the loss function w.r.t. epistemic uncertainty only (the uncertainty relevant to learning), whilst the aleatoric uncertainty remains free to be multimodal. We call this method distribution sampling (DS): $\boldsymbol{s}^p_{t+1} \sim \mathcal{N}\left(\mathbb{E}_b\left[\boldsymbol{s}^{p,b}_{t+1}\right], \mathbb{V}_b\left[\boldsymbol{s}^{p,b}_{t+1}\right]\right)$, with $\boldsymbol{s}^{p,b}_{t+1} \sim \widetilde{f}_{\boldsymbol{\theta}_b}(\boldsymbol{s}^p_t, \boldsymbol{a}_t)$.

6 Algorithm summary

Here we summarize our MBRL method *PETS* in Algorithm 1. We use the PE model to capture heteroskedastic aleatoric uncertainty and heteroskedastic epistemic uncertainty, which the TS planning method was able to best use. To guide the random shooting method of our MPC algorithm, we found that the CEM method learned faster (as discussed in Appendix A.8).

Algorithm 1 Our model-based MPC algorithm '*PETS*':

Initialize data $\mathbb{D}$ with a random controller for one trial.
for Trial $k = 1$ to K **do**
  Train a *PE* dynamics model $\widetilde{f}$ given $\mathbb{D}$.
  for Time $t = 0$ to TaskHorizon **do**
    for Actions sampled $\boldsymbol{a}_{t:t+T} \sim$ CEM$(\cdot)$, 1 to NSamples **do**
      Propagate state particles $\boldsymbol{s}^p_\tau$ using *TS* and $\widetilde{f}|\{\mathbb{D}, \boldsymbol{a}_{t:t+T}\}$.
      Evaluate actions as $\sum_{\tau=t}^{t+T} \frac{1}{P} \sum_{p=1}^{P} r(\boldsymbol{s}^p_\tau, \boldsymbol{a}_\tau)$
      Update CEM$(\cdot)$ distribution.
    Execute first action $\boldsymbol{a}^*_t$ (only) from optimal actions $\boldsymbol{a}^*_{t:t+T}$.
    Record outcome: $\mathbb{D} \leftarrow \mathbb{D} \cup \{\boldsymbol{s}_t, \boldsymbol{a}^*_t, \boldsymbol{s}_{t+1}\}$.

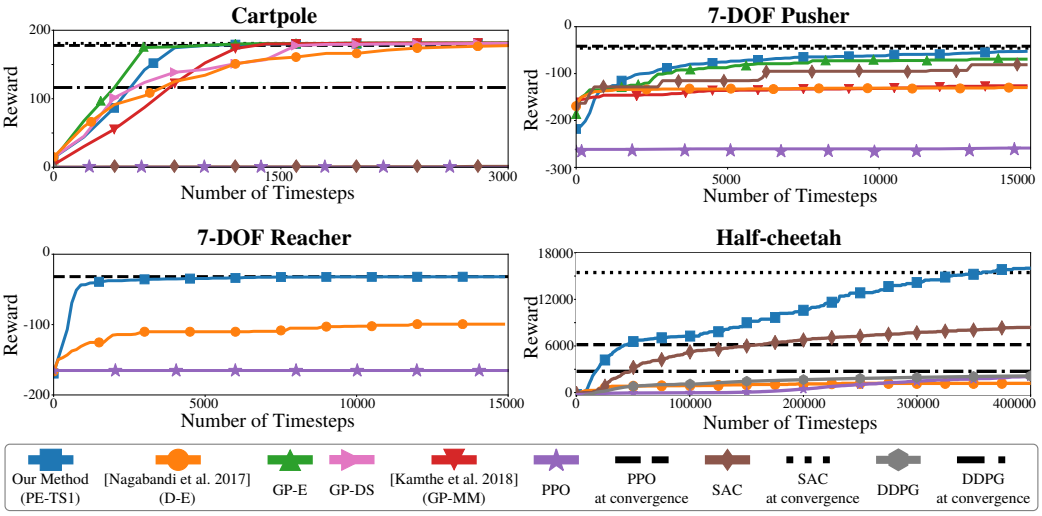
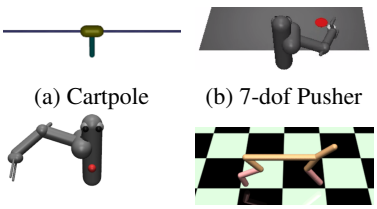


Figure 3: Learning curves for different tasks and algorithm. For all tasks, our algorithm learns in under 100K time steps or 100 trials. With the exception of Cartpole, which is sufficiently low-dimensional to efficiently learn a GP model, our proposed algorithm significantly outperform all other baselines. For each experiment, one time step equals 0.01 seconds, except Cartpole with 0.02 seconds. For visual clarity, we plot the average over 10 experiments of the maximum rewards seen so far.

7 Experimental results

We now evaluate the performance of our proposed MBRL algorithm called PETS using a deep neural network probabilistic dynamics model. First, we compare our approach on standard benchmark tasks against state-of-the-art model-free and model-based approaches in Section 7.1. Then, in Section 7.2, we provide a detailed evaluation of the individual design decisions in the model and uncertainty propagation method and analyze their effect on performance. Additional considerations of horizon length, action sampling distribution, and stochastic systems are discussed in Appendix A.7. The experiment setup is shown in Figure 2, and NN architecture details are discussed in the supplementary materials, in Appendix A.6. Videos of the experiments, and code for reproducing the experiments can be found at `https://sites.google.com/view/drl-in-a-handful-of-trials`.



(a) Cartpole (b) 7-dof Pusher

(c) 7-dof Reacher (d) Half-cheetah

Figure 2: Tasks evaluated.

7.1 Comparisons to prior reinforcement learning algorithms

We compare our Algorithm 1 against the following reinforcement learning algorithms for continuous state-action control:

- **Proximal policy optimization (PPO)**: [Schulman et al., 2017] is a model-free, deep policy-gradient RL algorithm (we used the implementation from Dhariwal et al. [2017].)
- **Deep deterministic policy gradient (DDPG)**: [Lillicrap et al., 2016] is an off-policy model-free deep actor-critic algorithm (we used the implementation from Dhariwal et al. [2017].)
- **Soft actor critic (SAC)**: [Haarnoja et al., 2018] is a model-free deep actor-critic algorithm, which reports better data-efficiency than DDPG on MuJoCo benchmarks (we obtained authors' data).
- **Model-based model-free hybrid (MBMF)**: [Nagabandi et al., 2017] is a recent deterministic deep model-based RL algorithm, which we reimplement.
- **Gaussian process dynamics model (GP)**: we compare against three MBRL algorithms based on GPs. GP-E learns a GP model, but only propagate the expectation. GP-DS uses the propagation method DS. GP-MM is the algorithm proposed by Kamthe and Deisenroth [2018] except that we do *not* update the dynamics model after each transition, but only at the end of each trial.

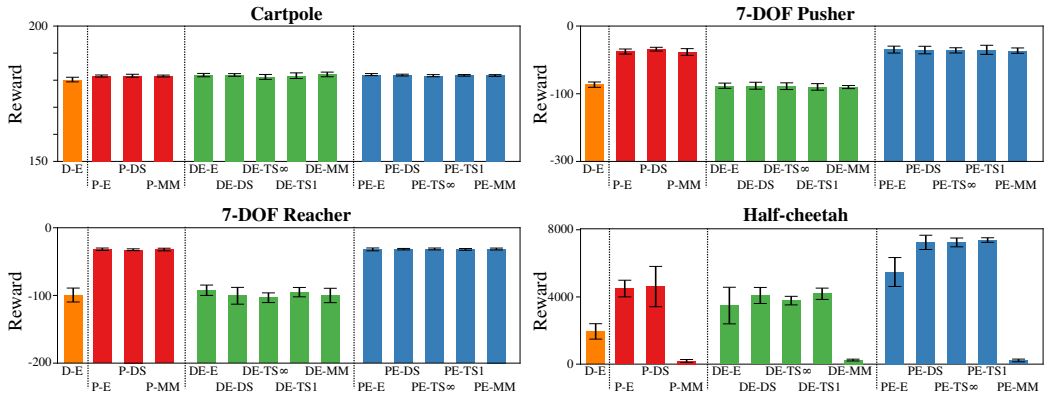


Figure 4: Final performance for different tasks, models, and uncertainty propagation techniques. The model choice seems to be more important than the technique used to propagate the state/action space. Among the models the ranking in terms of performance is: $PE > P > DE > D$. A linear model comparison can also be seen in Appendix A.10.

The results of the comparison are presented in Figure 3. Our method reaches performance that is similar to the asymptotic performance of the state-of-the-art model-free baseline PPO. However, PPO requires several orders of magnitude more samples to reach this point. We reach PPO's asymptotic performance in fewer than 100 trials on all four tasks, faster than any prior model-free algorithm, and the asymptotic performance substantially exceeds that of the prior MBRL algorithm by Nagabandi et al. [2017], which corresponds to the deterministic variant of our approach (D-E). This result highlights the value of uncertainty estimation. Whilst the probabilistic baseline GP-MM slightly outperformed our method in cartpole, GP-MM scales cubically in time and quadratically state dimensionality, so was infeasible to run on the remaining higher dimensional tasks. It is worth noting that model-based deep RL algorithms have typically been considered to be efficient but incapable of achieving similar asymptotic performance as their model-free counterparts. Our results demonstrate that a purely model-based deep RL algorithm that only learns a dynamics model, omitting even a parameterized policy, can achieve comparable performance when properly incorporating uncertainty estimation during modeling and planning. In the next section, we study which specific design decisions and components of our approach are important for achieving this level of performance.

7.2 Analyzing dynamics modeling and uncertainty propagation

In this section, we compare different choices for the dynamics model in Section 4 and uncertainty propagation technique in Section 5. The results in Figure 4 first show that w.r.t. model choice, the model should consider both uncertainty types: the probabilistic ensembles (PE-XX) perform best in all tasks, except cartpole ('X' symbolizes any character). Close seconds are the single-probability-type models: probabilistic network (P-XX) and ensembles of deterministic networks (E-XX). Worst is the deterministic network (D-E).

These observations shed some light on the role of uncertainty in MBRL, particularly as it relates to discriminatively trained, expressive parametric models such as NNs. Our results suggest that, the quality of the model and the use of uncertainty at learning time significantly affect the performance of the MBRL algorithms tested, while the use of more advanced uncertainty propagation techniques seem to offers only minor improvements. We reconfirm that moment matching (MM) is competitive in low-dimensional tasks (consistent with [Gal et al., 2016]), however is not a reliable MBRL choice in higher dimensions, e.g. the half cheetah.

The analysis provided in this section summarizes the experiments we conducted to design our algorithm. It is worth noting that the individual components of our method – ensembles, probabilistic networks, and various approximate uncertainty propagation techniques – have existed in various forms in supervised learning and RL. However, as our experiments here and in the previous section show, the particular choice of these components in our algorithm achieves substantially improved results over previous state-of-the-art model-based and model-free methods, experimentally confirming both the importance of uncertainty estimation in MBRL and the potential for MBRL to achieve asymptotic performance that is comparable to the best model-free methods at a fraction of the sample complexity.

8 Discussion & conclusion

Our experiments suggest several conclusions that are relevant for further investigation in model-based reinforcement learning. First, our results show that model-based reinforcement learning with neural network dynamics models can achieve results that are competitive not only with Bayesian nonparametric models such as GPs, but also on par with model-free algorithms such as PPO and SAC in terms of asymptotic performance, while attaining substantially more efficient convergence. Although the individual components of our model-based reinforcement learning algorithms are not individually new – prior works have suggested both ensembling and outputting Gaussian distribution parameters [Lakshminarayanan et al., 2017], as well as the use of MPC for model-based RL [Nagabandi et al., 2017] – the particular combination of these components into a model-based reinforcement learning algorithm is, to our knowledge, novel, and the results provide a new state-of-the-art for model-based reinforcement learning algorithms based on high-capacity parametric models such as neural networks. The systematic investigation in our experiments was a critical ingredient in determining the precise combination of these components that attains the best performance.

Our results indicate that the gap in asymptotic performance between model-based and model-free reinforcement learning can, at least in part, be bridged by incorporating uncertainty estimation into the model learning process. Our experiments further indicate that both epistemic and aleatoric uncertainty plays a crucial role in this process. Our analysis considers a model-based algorithm based on dynamics estimation and planning. A compelling alternative class of methods uses the model to train a parameterized policy [Ko et al., 2007, Deisenroth et al., 2014, McAllister and Rasmussen, 2017]. While the choice of using the model for planning versus policy learning is largely orthogonal to the other design choices, a promising direction for future work is to investigate how policy learning can be incorporated into our framework to amortize the cost of planning at test-time. Our initial experiments with policy learning did not yield an effective algorithm by directly propagating gradients through our uncertainty-aware models. We believe this may be due to chaotic policy gradients, whose recent analysis [Parmas et al., 2018] could help yield a policy-based PETS in future work. Finally, the observation that model-based RL can match the performance of model-free algorithms suggests that substantial further investigation of such of methods is in order, as a potential avenue for effective, sample-efficient, and practical general-purpose reinforcement learning.

References

P. Abbeel, M. Quigley, and A. Y. Ng. Using inaccurate models in reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 1–8, 2006. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143845.

P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. *Neural Information Processing Systems (NIPS)*, pages 5074–5082, 2016.

C. G. Atkeson and J. C. Santamaría. A comparison of direct and model-based reinforcement learning. In *International Conference on Robotics and Automation (ICRA)*, 1997.

A. Baranes and P.-Y. Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73, 2013. ISSN 0921-8890. doi: 10.1016/j.robot.2012.05.008.

R. Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, pages 679–684, 1957.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. *International Conference on Machine Learning (ICML)*, 37:1613–1622, 2015.

Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer. The cross-entropy method for optimization. In *Handbook of statistics*, volume 31, pages 35–59. Elsevier, 2013.

S. H. Brooks. A discussion of random methods for seeking maxima. *Operations Research*, 6(2): 244–251, 1958.

R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth. Manifold Gaussian processes for regression. In *International Joint Conference on Neural Networks (IJCNN)*, pages 3338–3345, 2016. doi: 10.1109/IJCNN.2016.7727626.

E. F. Camacho and C. B. Alba. *Model predictive control*. Springer Science & Business Media, 2013.

Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine. Combining model-based and model-free updates for trajectory-centric reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2017.

M. Deisenroth, D. Fox, and C. Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 37(2): 408–423, 2014. ISSN 0162-8828. doi: 10.1109/TPAMI.2013.218.

S. Depeweg, J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft. Learning and policy search in stochastic dynamical systems with Bayesian neural networks. *ArXiv e-prints*, May 2016.

S. Depeweg, J.-M. Hernandez-Lobato, F. Doshi-Velez, and S. Udluft. Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning. In *International Conference on Machine Learning (ICML)*, pages 1192–1201, 2018.

P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu. Openai baselines. `https://github.com/openai/baselines`, 2017.

A. Draeger, S. Engell, and H. Ranke. Model predictive control using neural networks. *IEEE Control Systems*, 15(5):61–66, Oct 1995. ISSN 1066-033X. doi: 10.1109/37.466261.

B. Efron and R. Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

C. Finn, X. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel. Deep spatial autoencoders for visuomotor learning. In *International Conference on Robotics and Automation (ICRA)*, 2016.

J. Fu, S. Levine, and P. Abbeel. One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4019–4026, 2016. doi: 10.1109/IROS.2016.7759592.

Y. Gal, R. McAllister, and C. Rasmussen. Improving PILCO with Bayesian neural network dynamics models. *ICML Workshop on Data-Efficient Machine Learning*, 2016.

Y. Gal, J. Hron, and A. Kendall. Concrete dropout. In *Neural Information Processing Systems (NIPS)*, pages 3584–3593, 2017.

A. Girard, C. E. Rasmussen, J. Quinonero-Candela, R. Murray-Smith, O. Winther, and J. Larsen. Multiple-step ahead prediction for non linear dynamic systems–a Gaussian process treatment with propagation of the uncertainty. *Neural Information Processing Systems (NIPS)*, 15:529–536, 2002.

A. Grancharova, J. Kocijan, and T. A. Johansen. Explicit stochastic predictive control of combustion plants based on Gaussian process models. *Automatica*, 44(6):1621–1631, 2008.

S. Gu, T. Lillicrap, I. Sutskever, and S. Levine. Continuous deep Q-learning with model-based acceleration. In *International Conference on Machine Learning (ICML)*, pages 2829–2838, 2016.

C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. *International Conference on Machine Learning (ICML)*, 2017.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, volume 80, pages 1856–1865, 2018.

E. Hernandaz and Y. Arkun. Neural network modeling and an extended DMC algorithm to control nonlinear systems. In *American Control Conference*, pages 2454–2459, May 1990.

J. M. Hernández-Lobato and R. Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

J. M. Hernández-Lobato, Y. Li, M. Rowland, D. Hernández-Lobato, T. Bui, and R. E. Turner. Black-box α-divergence minimization. *International Conference on Machine Learning (ICML)*, 48: 1511–1520, 2016.

J. C. G. Higuera, D. Meger, and G. Dudek. Synthesizing neural network controllers with probabilistic model based reinforcement learning. *arXiv preprint arXiv:1803.02291*, 2018.

S. Kamthe and M. P. Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

J. Ko, D. J. Klein, D. Fox, and D. Haehnel. Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 742–747. IEEE, 2007.

J. Kober and J. Peters. Policy search for motor primitives in robotics. In *Neural information processing systems (NIPS)*, pages 849–856, 2009.

J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and A. Girard. Gaussian process model based predictive control. In *American Control Conference*, volume 3, pages 2214–2219. IEEE, 2004.

A. G. Kupcsik, M. P. Deisenroth, J. Peters, and G. Neumann. Data-efficient generalization of robot skills with contextual policy search. In *Conference on Artificial Intelligence (AAAI)*, pages 1401–1407, 2013.

T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Neural Information Processing Systems (NIPS)*, pages 6405–6416. 2017.

I. Lenz, R. Knepper, and A. Saxena. DeepMPC: Learning deep latent features for model predictive control. In *Robotics Science and Systems (RSS)*, 2015.

S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*, 17(1):1334–1373, Jan. 2016. ISSN 1532-4435.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2016.

L.-J. Lin. *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis, Carnegie Mellon University, 1992.

D. J. MacKay. A practical Bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

R. McAllister and C. E. Rasmussen. Data-efficient reinforcement learning in continuous state-action Gaussian-POMDPs. In *Neural Information Processing Systems (NIPS)*, pages 2037–2046. 2017.

W. T. Miller, R. P. Hewes, F. H. Glanz, and L. G. Kraft. Real-time dynamic control of an industrial manipulator using a neural network-based learning controller. *IEEE Transactions on Robotics and Automation*, 6(1):1–9, Feb 1990. ISSN 1042-296X. doi: 10.1109/70.88112.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

I. Mordatch, N. Mishra, C. Eppner, and P. Abbeel. Combining model-based policy search with online model learning for control of physical humanoids. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 242–248, May 2016. doi: 10.1109/ICRA.2016.7487140.

A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *ArXiv e-prints*, Aug. 2017.

R. Neal. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995.

D. Nguyen-Tuong, J. Peters, and M. Seeger. Local Gaussian process regression for real time online model learning. In *Neural Information Processing Systems (NIPS)*, pages 1193–1200, 2008.

I. Osband. Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout. *NIPS Workshop on Bayesian Deep Learning*, 2016.

I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped DQN. In *Neural Information Processing Systems (NIPS)*, pages 4026–4034, 2016.

P. Parmas, C. E. Rasmussen, J. Peters, and K. Doya. PIPPS: Flexible model-based policy search robust to the curse of chaos. In *International Conference on Machine Learning (ICML)*, volume 80, pages 4062–4071, 2018.

A. Punjani and P. Abbeel. Deep learning helicopter dynamics models. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3223–3230, May 2015. doi: 10.1109/ICRA.2015.7139643.

J. Quiñonero-Candela, A. Girard, J. Larsen, and C. E. Rasmussen. Propagation of uncertainty in Bayesian kernel models—application to multiple-step ahead forecasting. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 701–704, April 2003. doi: 10.1109/ICASSP.2003.1202463.

P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. *CoRR*, abs/1710.05941, 2017. URL `http://arxiv.org/abs/1710.05941`.

C. E. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. In *Neural Information Processing Systems (NIPS)*, volume 4, page 1, 2003.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

S. Thrun. Efficient exploration in reinforcement learning. Technical Report CMU-CS-92-102, Carnegie Mellon University, 1992.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5026–5033, 2012.

G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou. Information theoretic MPC for model-based reinforcement learning. In *International Conference on Robotics and Automation (ICRA)*, 2017.

A Appendix

A.1 Well behaved probabilistic networks

An under-appreciated detail of probabilistic networks is how the variance output is implemented with automatic differentiation. Often the real-valued output is treated as a log variance (or similar), and transformed through an exponential function (or similar) to produce a nonnegative-valued output, necessary to be interpreted as a variance. However, whilst this variance output is well behaved at points within the training distribution, its value is undefined outside the trained distribution. In fact, during the training, there is no explicit loss term that regulate the behavior of the variance outside of the training points. Thus, when this model is then evaluated at previously unseen states, as is often the case during the MBRL learning process, the outputted variance can assume any arbitrary value, and in practice we noticed how it occasionally collapse to zero, or explode toward infinity.

This behavior is in contrast with other models, such as GPs, where the variance is more well behaving, being bounded and Lipschitz-smooth. As a remedy, we found that in our model lower bounding and upper bounding the output variance such that they could not be lower or higher than the lowest and highest values in the training data significantly helped. To bound the variance output for a probabilistic network to be between the upper and lower bounds found during training the network on the training data, we used the following code with automatic differentiation:

```
logvar = max_logvar - tf.nn.softplus(max_logvar - logvar)
logvar = min_logvar + tf.nn.softplus(logvar - min_logvar)
var = tf.exp(logvar)
```

with a small regularization penalty on term on `max_logvar` so that it does not grow beyond the training distribution's maximum output variance, and on the negative of `min_logvar` so that it does not drop below the training distribution's minimum output variance.

A.2 Fitting PE model to toy function

As an initial test, we evaluated all previously described models by fitting to a dataset $\{(x_i, y_i)\}$ of 2000 points from a sine function, where the x_i's are sampled uniformly from $[-2\pi, -\pi] \cup [\pi, 2\pi]$. Before fitting, we introduced heteroscedastic noise by performing the transformation

$$(x, y) \mapsto \left(x, y + \mathcal{N}\left(0, 0.0225 \left|\sin\left(\frac{3}{2}x + \frac{\pi}{8}\right)\right|\right)\right). \tag{2}$$

The model fit to (2) was shown in Figure 1, but reproduced here for convenience as Figure A.5.

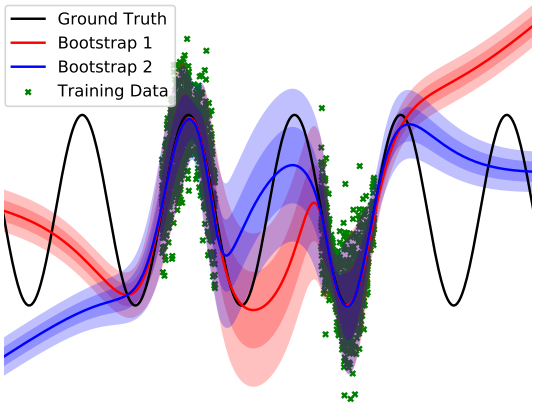
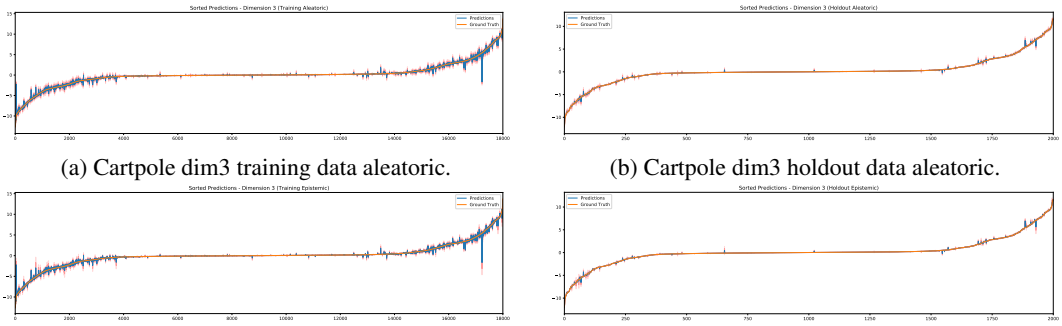


Figure A.5: Our probabilistic ensemble (PE) dynamics model: an ensemble of two bootstraps (for visual clarity, we normally use five bootstraps), each a probabilistic neural network that captures aleatoric uncertainty (in this case: observation noise). Note the bootstraps agree near data, but tend to disagree far from data. Such bootstrap disagreement represents our model's epistemic uncertainty.

A.3 One-step predictions of learned models

To visualize and verify the accuracy of our PE model, we took all training data from the experiments and visualized the one-step predictions of the model. Since the states are high-dimensional, we resorted to plotting the output dimensions individually, sorting by the ground truth value in each dimension, seen in Figure A.6.



(a) Cartpole dim3 training data aleatoric.

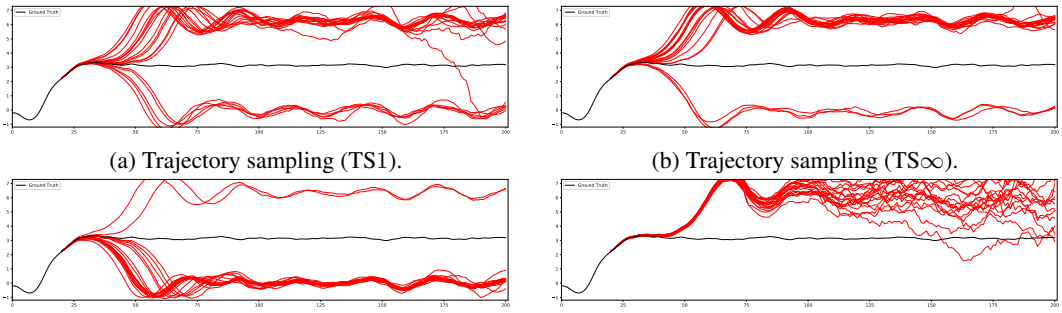
(b) Cartpole dim3 holdout data aleatoric.

(c) Cartpole dim3 training data epistemic.

(d) Cartpole dim3 holdout data epistemic.

Figure A.6: One step predictions of the cartpole angular velocity (velocities are typically harder to predict) after 100 trails of training data. Shown are the prediction indexes, monotonically increase in ground truth output value, with two standard deviations at each output prediction. We see the model is certain (w.r.t. both uncertainty types) where most of the data lies, but less certain in extreme values of data where there are fewer training data.

A.4 Uncertainty propagation methods



(a) Trajectory sampling (TS1).

(b) Trajectory sampling (TS∞).

(c) Distribution sampling (DS).

(d) Moment matching (MM).

Figure A.7: Different uncertainty propagation methods discussed in Section 5. We show a PE model trained after 100 trials on the cartpole system propagating particles given an action sequence from an intermediate state (pole swinging up) that solves the task.

A.5 Forward Dynamics Model

Following the suggestion presented in [Deisenroth et al., 2014], instead of learning a forward dynamics in the form $\boldsymbol{s}_{t+1} = f\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)$, we learn a model that predicts the difference to the current state $\Delta \boldsymbol{s}_{t+1} = f\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)$ such that $\boldsymbol{s}_{t+1} = \boldsymbol{s}_t + \Delta \boldsymbol{s}_{t+1}$. Moreover, for states $\boldsymbol{s}_i$ that represent angles, we transform the states fed as inputs to the dynamics model to be $[\sin(\boldsymbol{s}_i), \cos(\boldsymbol{s}_i)]$ to capture the rotational nature of the joint.

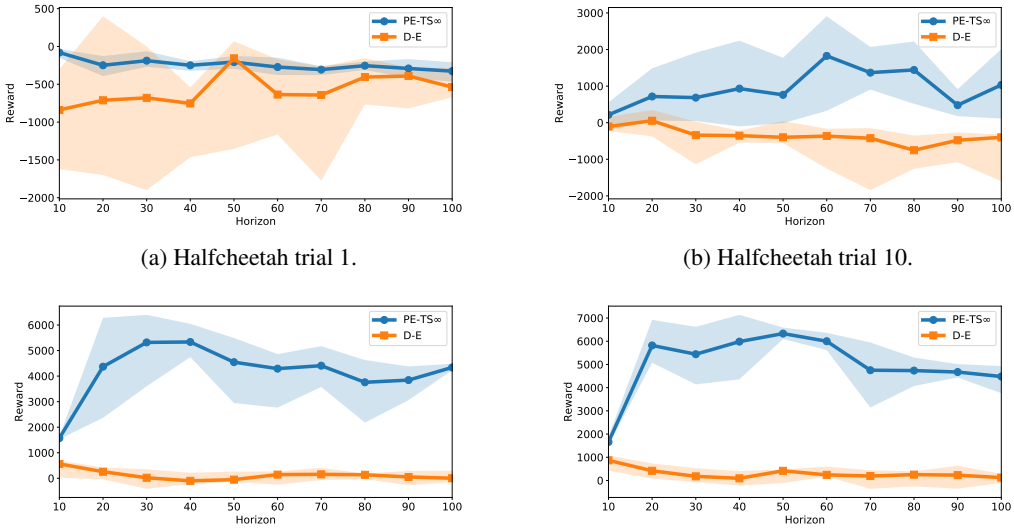
A.6 Experimental setting

For our experiments, we used four continuous-control benchmark tasks simulated via MuJoCo [Todorov et al., 2012] that vary in complexity, dimensionality, and the presence of contact forces (pictured Figure 2). The simplest is the classical cartpole swing-up benchmark ($d_s = 4$, $d_a = 1$). To

evaluate our model with higher dimensional dynamics and frictional contacts, we use a simulated PR2 robot in a reaching and pushing task ($d_s = 14$, $d_a = 7$), as well as the half-cheetah ($d_s = 17$, $d_a = 6$). Each experiment is repeated with different random seeds, and the mean and standard deviation of the cost is reported for each condition. Each neural network dynamics model consist of three fully connected layers, 500 neurons per layer (except 250 for halfcheetah), and swish activation functions [Ramachandran et al., 2017]. The weights of the networks were initially sampled from a truncated Gaussian with variance equal to the reciprocal of the number of fan-in neurons.

A.7 Additional considerations

MPC horizon length: choosing the MPC horizon T is nontrivial: 'too short' and MPC suffer from bias, 'too long' then variance. Probabilistic propagation methods are robust to horizons set 'too long'. This effect is due to particle separation over time (e.g. Figure A.7), which reduces the dependence of actions on *expected*-cost further in time. The action selection procedure then effectively ignores the unpredictable with our method. Deterministic methods have no such mechanism to avoid model bias [Deisenroth et al., 2014], which compounds over longer time horizons, resulting in poor performance if the horizon is set 'too high' as seen in Figure A.8.



(a) Halfcheetah trial 1.

(b) Halfcheetah trial 10.

(c) Halfcheetah trial 40.

(d) Halfcheetah trial 80.

Figure A.8: Effect of MPC horizon on halfcheetah after different amounts of trials. Showing median, and percentile bound 5 and 95, from 5 repeats of experiment.

MPC action sampling: We hypothesized the higher the state or action dimensionality, the more important that MPC action selection is guided (opposed to the uniform random shooting method, used by Nagabandi et al. [2017]). Thus we tested cross-entropy method (CEM) and random shooting for various tasks confirming this hypothesis (details Appendix A.8).

Stochastic systems: Finally we evaluate how successful probabilistic networks mitigate the detrimental effects of system stochasticity whilst learning to control. We introduced probabilistic networks as a means of capturing aleatoric uncertainty (inherent and persistent system stochasticities). Here we test how well probabilistic networks perform against deterministic networks under stochasticities in the action space. We add Gaussian noise onto the robot's selected action, of standard deviations ranging 0-20% of action ranges permitted by MuJoCo. Figure A.9 shows that probabilistic PE models perform better and more consistently under system noise. Further visualizations are provided in Appendix A.9.

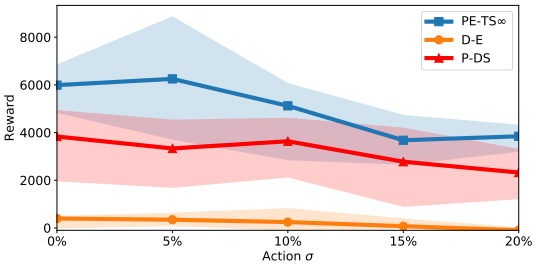
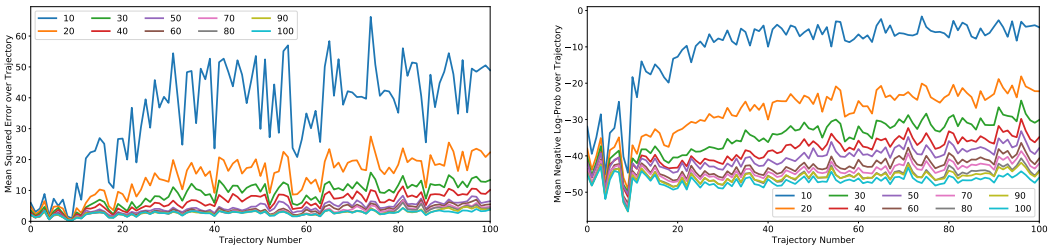


Figure A.9: Modeling aleatoric uncertainty makes MBRL more robust to stochasticity.

Model accuracy over time: Figure A.10 shows the evolution of a PE model's accuracy on the halfcheetah as it collects model trails of data (see legend).



(a) Mean squared error.

(b) Negative log likelihood.

Figure A.10: Model accuracy: our PETS dynamics model at trials 10-100 (see legend) make predictions on trajectory seen at each trial (x-axis) and are scored (y-axis) according to mean squared error (left figure) and negative log likelihood (right figure).

A.8 MPC action selection

We study the impact of the particular choice of action optimization technique. An important criterion when selecting the optimizer is not only the optimality of the selected actions, but also the speed with which the actions can be obtained, which is especially critical for real-world control tasks that must proceed in real time[2]. Simple random search techniques have been proposed in prior work due to their simplicity and ease of parallelism [Nagabandi et al., 2017]. However, uniform random search [Brooks, 1958] suffers in high-dimensional spaces. In addition to random search, we compare to the cross-entropy method (CEM) [Botev et al., 2013], which iteratively samples solutions from a candidate distribution that is adjusted based on the best sampled solutions. To isolate the comparison of optimizers from our dynamics model, we instead use the ground truth dynamics function (the MuJoCo simulator itself) to evaluate candidate action sequences. The results (Figure A.11) show that using CEM significantly outperforms random search on the half-cheetah task. We use CEM in all of the remaining experiments.

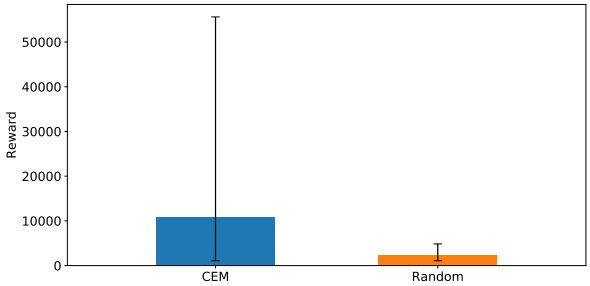
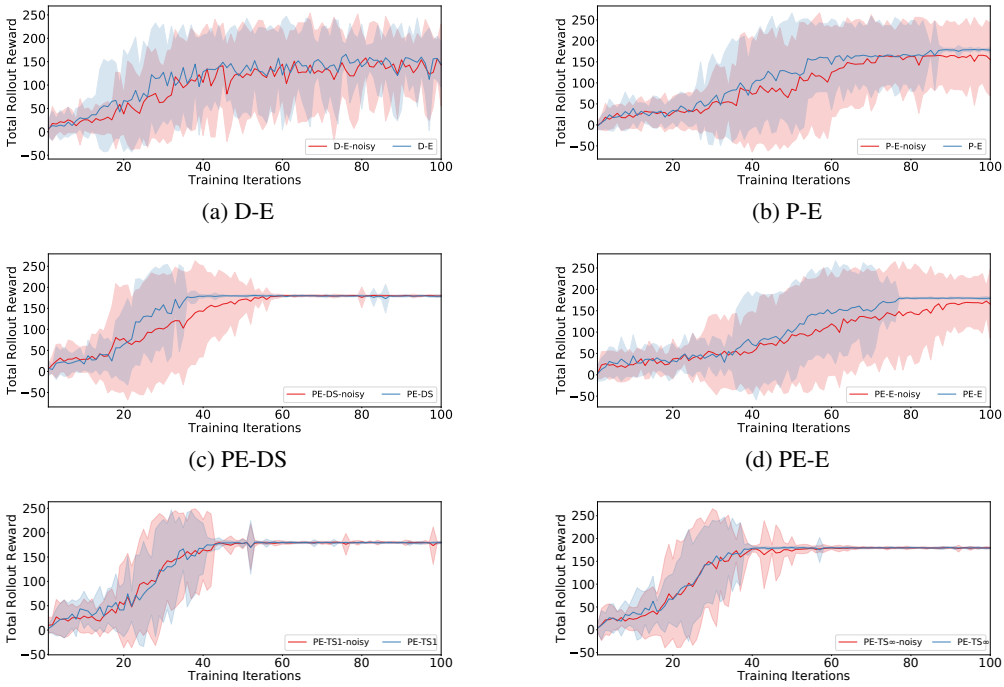


Figure A.11: Average reward achieved on ground truth dynamics of the half-cheetah (using the MuJoCo simulator itself as ground truth dynamics). The cross entropy method (CEM) optimizer performs significantly better than random shooting sampling. For fair comparison, both use 2500 samples: CEM has five iterations of sampling 500 candidate actions before choosing the elite candidates, whereas random shooting simply sampled 2500 times. Shown is the median performance, with error bars showing the 5 and 95 percentile performance across random seeds.

[2]Such as robotics, where control frequencies below 20Hz are undesirable, meaning that a decision need to be taken in under 50ms.

A.9 Stochastic systems:

In Figure A.12f we compare and contrast the effect stochastic action noise has w.r.t. variable MBRL modeling decisions. Notice methods that PE method that propagate uncertainty are generally required for *consistent* performance.



(a) D-E (b) P-E

(c) PE-DS (d) PE-E

(e) PE-TS1 (f) PE-TS∞

Figure A.12: The distribution of cartpole's reward for particular MBRL design decisions in the presence of stochastic system noise (in this case additive noise onto the actions selected by the robot: with standard deviation equal to 10% of each of the action range.)

A.10 Linear model comparison:

Figure A.13 shows that a linear model is unable to capture the halfcheetah dynamics well enough to control it, and that a nonlinear model is necessary.

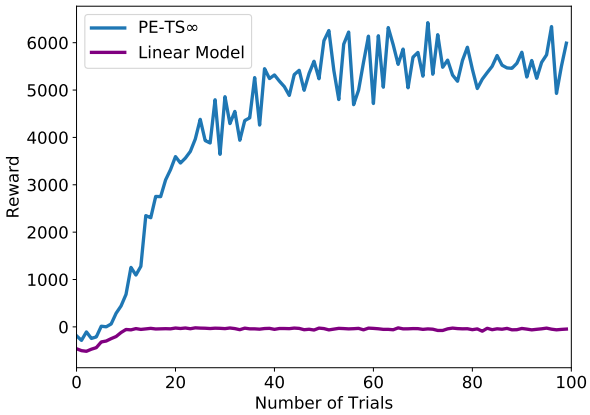


Figure A.13: Linear model comparison.