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

Kurtland Chua, Roberto Calandra, Rowan McAllister, Sergey Levine
University of California, Berkeley
{kchua,roberto.calandra,rmcallister}@berkeley.edu,svlevine@eecs.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, especially those 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. 25 and 125 times fewer samples than Soft Actor Critic and Proximal Policy Optimization respectively on the half-cheetah task).**

I. 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 [1], [2], [3], with a few exceptions [4], [5]. A promising direction to 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 [6], [7], [8]. 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 reward 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 more suboptimal 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 [9]. Neural network (NN) models can scale to large datasets with high-dimensional inputs, and can represent such systems more effectively. However, they struggle with the opposite problem: in the low-data regime in which MBRL always starts, they tend to overfit and make 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 [10], [11], including in the context of RL [12], [13], our work is, to our knowledge, the first to bring these components together in a MBRL framework that approaches the asymptotic performance of state-of-the-art model-free RL methods, at a fraction of the sample complexity.

Our main contribution is a MBRL algorithm called probabilistic ensembles with trajectory sampling (PETS) summarized in Figure 1) with high-capacity NN models that incorporate uncertainty via an ensemble of bootstrapped models, where each model encodes *distributions* (opposed to point predictions), rivaling the performance model-free methods on standard benchmark tasks at a fraction of the sample complexity. An additional 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 [14], although we leave this for future work. Finally, we present a systematic analysis of how incorporating uncertainty into MBRL with NNs affects performance, in both model training and planning. We show, that PETS's particular treatment of uncertainty significantly reduces the amount of data required to learn a task, e.g. 25 times fewer data on half-cheetah compared to the model-free Soft Actor Critic algorithm [15].

Accepted at the IROS 2018 Workshop on Machine Learning in Robot Motion Planning.

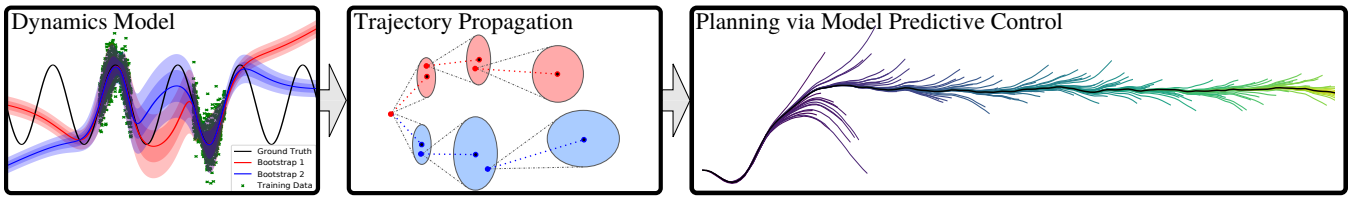

Fig. 1: Our method (PE-TS): **Model**: Our probabilistic ensemble (PE) dynamics model is shown as an ensemble of two bootstraps (bootstrap disagreement far from data captures epistemic uncertainty), each a probabilistic neural network that captures aleatoric uncertainty (inherent variance of the observed 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 MPC-horizon. **Planning**: Our MPC algorithm recompute at each time-step and applies the first action of each action sequence, up until the task-horizon.

II. 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 GPs, are often the model of choice in MBRL [7], [16], [17], [18], [8], [19]. However, such models typically induce additional assumptions on the system, such as the smoothness assumption inherent in GPs with squared-exponential kernels [20]. Parametric function approximators have also been used extensively in MBRL [21], [22], [23], [24], but were largely supplanted by Bayesian models in recent years. Methods based on local models, such as guided policy search algorithms [5], [25], [26], can efficiently train NN policies, but using 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 [27], [28], [29], [30], [31], [12], [13], [32], [33]. Unlike Gaussian processes, NNs have constant-time inference and tractable training in the large data regime, and have potential to represent more complex functions, including non-smooth dynamics that are often present in robotics [28], [34], [33]. 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 that prior deterministic MBRL methods such as [33].

Constructing good Bayesian NN models remains an open problem [35], [10], [36], [37], although recent promising work exists on incorporating dropout [38], ensembles [39], [11], and α-divergence [40]. Such probabilistic NNs have previously been used for control, including using dropout [12], [41] and α-divergence [13]. 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.

III. Model-based reinforcement learning

We now detail the MBRL framework and the notation used. Adhering to the Markov decision process formulation [42], 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: $f_{\boldsymbol{\theta}}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) = \Pr(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$, 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 it to predict the distribution over trajectories resulting from applying a given policy (e.g., a sequence of actions). We hence compute the distribution of the reward for a given policy, and use it to optimize the best policy to use. In Section IV we discuss multiple methods to model the dynamics, while in Section V we detail how to compute the distribution over trajectories and how to parametrize the controller.

IV. Uncertainty-aware neural network dynamics models

This section describes a number of ways to model uncertain dynamics, including our method: an ensemble of bootstrapped probabilistic neural networks. Whilst uncertainty-aware dynamics models have been explored in a number of prior works [12], [13], 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 [33], and often even found that simpler time-varying linear models can outperform expressive neural network models [5], [43].

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 [6], [44]. A major challenge 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 [9], which is typical of robots interacting through contacts.

Fig. 2: 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**

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 2.

a) Probabilistic neural networks (P): We define a *probabilistic* NN as a network whose output neurons encode any parametric distribution, capturing aleatoric uncertainty. 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 distributions with diagonal covariances parameterized by $\boldsymbol{\theta}$ and conditioned on $\boldsymbol{s}_n$ and $\boldsymbol{a}_n$, i.e.: $\widetilde{f} = \Pr(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) = \mathcal{N}\left(\boldsymbol{\mu}_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t), \boldsymbol{\Sigma}_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t)\right)$. We can derive a loss function from the log-likelihood. Such a network output, which parameterizes a Gaussian distribution, models heteroscedastic aleatoric uncertainty (heteroscedasticity means the random output variability 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.

b) 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).

c) 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 [10], approximate inference methods [45], [38], [46] 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 [47], [36]. 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 [11] – both of which define predictive probability distributions: $\widetilde{f}_{\boldsymbol{\theta}} = \frac{1}{B}\sum_{b=1}^{B} \widetilde{f}_{\boldsymbol{\theta}_b}$. 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.

V. 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, the choice of action $\boldsymbol{a}_t$ can depend on the state, forming a policy, $\pi : \boldsymbol{s}_t \to \boldsymbol{a}_t$. Otherwise, planning with actions independent of the state is typically framed as model predictive control (MPC) [48]. MPC can be considered a special-case policy, trained as a function of the dynamics model, and thereafter only dependent on time, not state. We use MPC in our own experiments for several reasons, including implementation simplicity,

lower computational burden (no gradients), no requirement to specify the task-horizon in advance, whilst achieving the same data-efficiency as [12] 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 VI.

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. [33] use deterministic NN models and MPC with random shooting to achieve data efficient control for in higher dimensional tasks than what is feasible for GP to model. Our work improves upon [33]'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 [49] instead of random-shooting, which samples actions from a distribution closer to previous action samples that yielded high reward.

Evaluating the exact expected trajectory reward using recursive state prediction is generally intractable. Multiple approaches to approximate uncertainty propagation found in the literature [50], [51]. 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 [8], experimental results [52] 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 V-A. 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 V-B.

A. Our state propagation method: trajectory sampling (TS)

Our method to predict plausible state trajectories begins by creating P particles from the current state, $\boldsymbol{s}^p_{t=0} = \boldsymbol{s}_0 \, \forall \, p$. We found $P = 20$ sufficient in all our experiments. For each particle we associate a bootstrap $b(p, t)$, sampled uniformly from $\{1, \ldots, B\}$, 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. Since 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 aleatoric and epistemic uncertainties are separable (e.g. 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 [14]. 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.

In both variants, we then propagate particles by sampling $\boldsymbol{s}^p_{t+1} \sim \widetilde{f}_{\boldsymbol{\theta}_{b(p,t)}}(\boldsymbol{s}^p_t, \boldsymbol{a}_t)$. Note that TS can capture multi-modal distributions and can be used with any probabilistic model.

B. Baseline state propagation methods for comparison

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

a) 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 small model biases can compound quickly over time, with no way to tell the quality of the state estimate.

b) 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 [12]. Although unclear why, [12] (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.

c) 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): $\boldsymbol{s}_{t+1}^p \sim \mathcal{N}\left(\mathbb{E}_b\left[\boldsymbol{s}_{t+1}^{p,b}\right], \mathbb{V}_b\left[\boldsymbol{s}_{t+1}^{p,b}\right]\right)$, with $\boldsymbol{s}_{t+1}^{p,b} \sim \widetilde{f}_{\boldsymbol{\theta}_b}(\boldsymbol{s}_t^p, \boldsymbol{a}_t)$.

VI. Algorithm summary

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 $\boldsymbol{s}_{\tau}^p$ using *TS* and $\widetilde{f}|\{\mathbb{D}, \boldsymbol{a}_{t:t+T}\}$.
      Evaluate actions as $\sum_{\tau=t}^{t+T} \frac{1}{P} \sum_{p=1}^{P} r(\boldsymbol{s}_{\tau}^p, \boldsymbol{a}_{\tau})$
      Update CEM$(\cdot)$ distribution.
    Execute first action $\boldsymbol{a}_t^*$ (only) from optimal actions $\boldsymbol{a}_{t:t+T}^*$.
    Record outcome: $\mathbb{D} \leftarrow \mathbb{D} \cup \{\boldsymbol{s}_t, \boldsymbol{a}_t^*, \boldsymbol{s}_{t+1}\}$.

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 the CEM method learned faster.

VII. Experimental results

We now evaluate the performance of our proposed MBRL algorithm called PETS using deep probabilistic dynamics models. First, we compare our approach on standard benchmark tasks against state-of-the-art model-free and model-based approaches in Section VII-A. Then, in Section VII-B, we provide a detailed evaluation of the individual design decisions in the model and uncertainty propagation method and analyze their effect on performance. Experiment videos as well as a link to experiment code can be found at `https://sites.google.com/view/drl-in-a-handful-of-trials`.

A. 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)**: [3] is a model-free, deep policy-gradient RL a algorithm (we used the implementation from [53].)
- **Deep deterministic policy gradient (DDPG)**: [2] is an off-policy model-free deep actor-critic algorithm (we used the implementation from [53].)
- **Soft actor critic (SAC)**: [15] 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)**: [33] 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 [54] except that we do *not* update the dynamics model after each transition, but only at the end of each trial.

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 [33] 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.

B. Analyzing dynamics modeling and uncertainty propagation

In this section, we compare different choices for the dynamics model in Section IV and uncertainty propagation technique in Section V. 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 effect the performance of the MBRL algorithms tested, while the use of more aadvanced uncertainty propagation technique seem to offers only minor improvements. We reconfirm that moment matching (MM) is competitive in low-dimensional tasks (consistent with [12]), however is not a reliable MBRL choice in higher dimensions, e.g. the half cheetah.

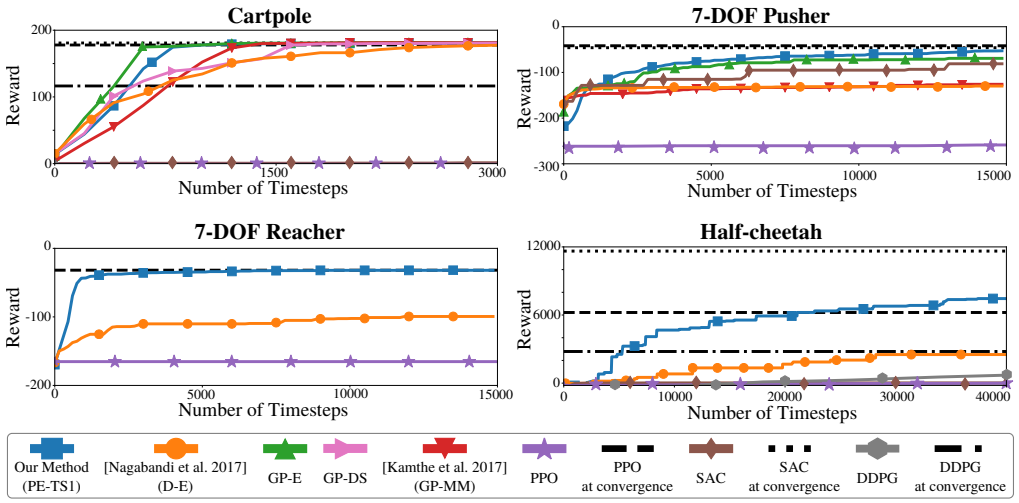

Fig. 3: Learning curves for different tasks and algorithm. For all tasks, our algorithm learns in under 100K time steps or 100 trials. With the exception of Cartpole, which is sufficiently low-dimensional to efficiently learn a GP model, our proposed algorithm significantly outperform all other baselines. For visual clarity, we plot the maximum reward seen so far, averaged over 10 experiments.

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 comparable to the best model-free methods at a fraction of the sample complexity.

VIII. 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 [11], as well as the use of MPC for model-based RL [33] – 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.

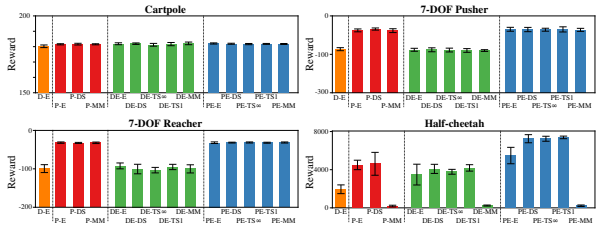

Fig. 4: Final performance for different tasks, models, and uncertainty propagation techniques. The model choice seems to be more important than the technique used to propagate the state/action space. Among the models the ranking in terms of performance is: $PE > P > DE > D$.

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 model-based algorithm based on dynamics estimation and planning. A compelling alternative class of methods uses the model to train a parameterized policy [16], [8], [55]. 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, though future work could consider alternative methods for policy learning. 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

[1] 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*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] 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," *arXiv preprint arXiv:1509.02971*, 2015.

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

[4] J. Kober and J. Peters, "Policy search for motor primitives in robotics," in *Advances in neural information processing systems (NIPS)*, pp. 849–856, 2009.

[5] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, pp. 1334–1373, Jan. 2016.

[6] C. G. Atkeson and J. C. Santamaría, "A comparison of direct and model-based reinforcement learning," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 1997.

[7] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and A. Girard, "Gaussian process model based predictive control," in *American Control Conference*, vol. 3, pp. 2214–2219, IEEE, 2004.

[8] 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)*, vol. 37, no. 2, pp. 408–423, 2014.

[9] R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth, "Manifold Gaussian processes for regression," in *International Joint Conference on Neural Networks (IJCNN)*, pp. 3338–3345, 2016.

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

[11] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Neural Information Processing Systems (NIPS)*, pp. 6405–6416, 2017.

[12] Y. Gal, R. McAllister, and C. Rasmussen., "Improving PILCO with Bayesian neural network dynamics models," 2016.

[13] 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.

[14] S. Thrun, "Efficient exploration in reinforcement learning," 1992.

[15] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.

[16] 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)*, pp. 742–747, IEEE, 2007.

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

[18] A. Grancharova, J. Kocijan, and T. A. Johansen, "Explicit stochastic predictive control of combustion plants based on Gaussian process models," *Automatica*, vol. 44, no. 6, pp. 1621–1631, 2008.

[19] 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.

[20] C. E. Rasmussen, M. Kuss, *et al.*, "Gaussian processes in reinforcement learning.," in *Neural Information Processing Systems (NIPS)*, vol. 4, p. 1, 2003.

[21] E. Hernandaz and Y. Arkun, "Neural network modeling and an extended DMC algorithm to control nonlinear systems," in *1990 American Control Conference*, pp. 2454–2459, May 1990.

[22] 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*, vol. 6, pp. 1–9, Feb 1990.

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

[24] A. Draeger, S. Engell, and H. Ranke, "Model predictive control using neural networks," *IEEE Control Systems*, vol. 15, pp. 61–66, Oct 1995.

[25] 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.

[26] 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.

[27] A. Baranes and P.-Y. Oudeyer, "Active learning of inverse models with intrinsically motivated goal exploration in robots," *Robotics and Autonomous Systems*, vol. 61, no. 1, pp. 49–73, 2013.

[28] J. Fu, S. Levine, and P. Abbeel, "One-shot learning of manipulation skills with online dynamics adaptation and neural network priors," *arXiv preprint*, 2015. arXiv:1509.06841.

[29] A. Punjani and P. Abbeel, "Deep learning helicopter dynamics models," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3223–3230, May 2015.

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

[31] P. Agrawal, A. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to poke by poking: Experiential learning of intuitive physics," *arXiv preprint*, 2016. arXiv:1606.07419.

[32] 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.

[33] 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.

[34] 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)*, pp. 242–248, May 2016.

[35] D. J. MacKay, "A practical Bayesian framework for backpropagation networks," *Neural computation*, vol. 4, no. 3, pp. 448–472, 1992.

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

[37] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," *arXiv preprint arXiv:1706.04599*, 2017.

[38] Y. Gal, J. Hron, and A. Kendall, "Concrete dropout," in *Advances in Neural Information Processing Systems*, pp. 3584–3593, 2017.

[39] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped DQN," in *Neural Information Processing Systems (NIPS)*, pp. 4026–4034, 2016.

[40] J. M. Hernández-Lobato, Y. Li, M. Rowland, D. Hernández-Lobato, T. Bui, and R. E. Turner, "Black-box α-divergence minimization," 2016.

[41] 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.

[42] R. Bellman, "A Markovian decision process," *Journal of Mathematics and Mechanics*, pp. 679–684, 1957.

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

[44] P. Abbeel, M. Quigley, and A. Y. Ng, "Using inaccurate models in reinforcement learning," in *International Conference on Machine Learning (ICML)*, pp. 1–8, 2006.

[45] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," *arXiv preprint arXiv:1505.05424*, 2015.

[46] J. M. Hernández-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of Bayesian neural networks," in *International Conference on Machine Learning*, pp. 1861–1869, 2015.

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

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

[49] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer, "The cross-entropy method for optimization," in *Handbook of statistics*, vol. 31, pp. 35–59, Elsevier, 2013.

[50] 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)*, vol. 15, pp. 529–536, 2002.

[51] 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*, vol. 2, pp. 701–704, April 2003.

[52] A. G. Kupcsik, M. P. Deisenroth, J. Peters, G. Neumann, *et al.*, "Data-efficient generalization of robot skills with contextual policy search," in *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, pp. 1401–1407, 2013.

[53] 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.

[54] S. Kamthe and M. P. Deisenroth, "Data-efficient reinforcement learning with probabilistic model predictive control," *CoRR*, vol. abs/1706.06491, 2017.

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