Variational Inference MPC for Bayesian Model-based Reinforcement Learning

Masashi Okada
Panasonic Corp., Japan
okada.masashi001@jp.panasonic.com

Tadahiro Taniguchi
Ritsumeikan Univ. & Panasonic Corp., Japan
taniguchi@em.ci.ritsumei.ac.jp

Abstract: In recent studies on model-based reinforcement learning (MBRL), incorporating uncertainty in forward dynamics is a state-of-the-art strategy to enhance learning performance, making MBRLs competitive to cutting-edge model-free methods, especially in simulated robotics tasks. Probabilistic ensembles with trajectory sampling (PETS) is a leading type of MBRL, which employs Bayesian inference to dynamics modeling and model predictive control (MPC) with stochastic optimization via the cross entropy method (CEM). In this paper, we propose a novel extension to the uncertainty-aware MBRL. Our main contributions are twofold: Firstly, we introduce a variational inference MPC (VI-MPC), which reformulates various stochastic methods, including CEM, in a Bayesian fashion. Secondly, we propose a novel instance of the framework, called probabilistic action ensembles with trajectory sampling (PaETS). As a result, our Bayesian MBRL can involve multimodal uncertainties both in dynamics and optimal trajectories. In comparison to PETS, our method consistently improves asymptotic performance on several challenging locomotion tasks.

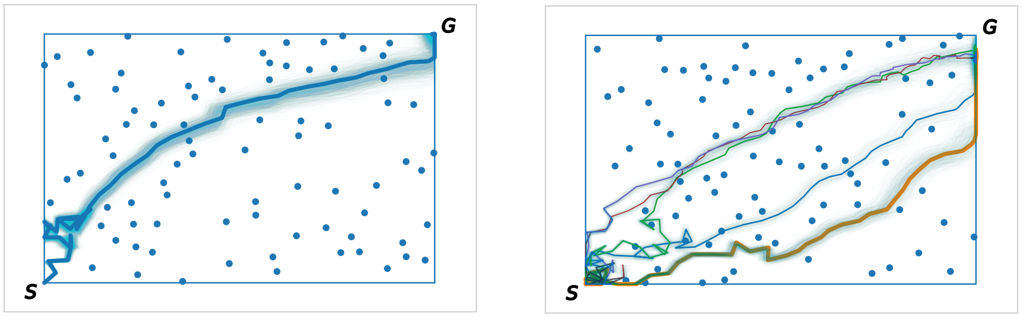
Keywords: model predictive control, variational inference, model-based reinforcement learning

1 Introduction

Model predictive control (MPC) is a powerful and accepted technology for advanced control systems such as manufacturing processes [1], HVAC systems [2], power electronics [3], autonomous vehicles [4], and humanoids [5]. MPC utilizes the specified models of system dynamics to predict future states and rewards (or costs) to plan future actions that maximize the total reward over the predicted trajectories. Especially for industrial applications, the clear explainability of such a decision-making process is advantageous. Furthermore, in some tasks (e.g., games) [6], planning-based policies of this nature could outperform *reactive*-policies (e.g., full neural network policies).

Model-based reinforcement learning (MBRL) methods that employ expressive function approximators (e.g., deep neural networks: DNNs) [7, 8, 9] present appealing approaches for MPC. The main difficulty in introducing MPC to practical systems is specifying the forward dynamics models of target systems. However, accurate system identification is challenging in many advanced applications. Take robotics for example, where robots encounter floors and walls, and must be able to manipulate some objects, making the dynamics highly non-linear. The main objective of MBRL is to train approximators of complex dynamics through experiences in real systems. The general procedure of MBRL is summarized as; (1. *training-step*) train the approximate model with a given training dataset, then (2. *test-step*) execute the actions (or policies) optimized with the dynamics model in a real environment and augment the dataset with the observed results. The above training and test steps are iteratively conducted to collect sufficient and diverse data so as to achieve the desired performance.

One feature of MBRL is its considerable sample efficiency compared to model-free reinforcement learning (MFRL), which directly trains policies through experiences. In other words, MBRL requires much less test time in real environments. In addition, MBRL benefits from the generalizability of the trained model, which can be easily applied to new tasks in the same system. However, the



(a) Vanilla CEM used in PETS [13]: `VIMPC('CEM', 'Gaussian', False)`

(b) PaETS (Ours): `VIMPC('CEM', 'GMM(M=5)', True)`

Figure 1: Toy task examples that illustrate the concept of our method. The objective of this task is to navigate a point mass on the x-y plane by actuating $\boldsymbol{a}_t = (\Delta x, \Delta y)$ with maximum magnitude $||\boldsymbol{a}_t|| = 0.05$, while avoiding obstacles ●. This task is designed to have multiple (sub-)optimal trajectories. (a) A trajectory found by vanilla CEM. (b) Multiple trajectories found by PaETS that approximates the trajectory posterior via variational inference with a Gaussian mixture model. The line-width indicates the magnitude of mixture-coefficients. Exploiting diverse plans encourages active exploration in state-action spaces, improving the optimization performance and training dataset diversity. The notation of `VIMPC()` is introduced in Sec. 3.

asymptotic performance of MBRL is generally inferior to that of model-free methods. This discrepancy is primarily due to the overfitting of dynamics models to the few data available during initial MBRL steps, which is called the *model-bias problem* [7]. Several studies have demonstrated that incorporating uncertainty in dynamics models can alleviate this issue. The uncertainty-aware modeling is realized by Bayesian inference employing a Gaussian Process [7], *dropout as variational inference* [10, 11, 12], or neural network ensembles [13, 14, 15].

Probabilistic ensembles with trajectory sampling (PETS) [13] is one type of uncertainty-aware MBRL. As an MPC-oriented MBRL method, PETS conducts trajectory optimization via the cross entropy method (CEM) [16] by using trajectories probabilistically sampled from the ensemble networks. Experiments have demonstrated that PETS can achieve competitive performance over state-of-the-art MFRL methods like Soft Actor Critic (SAC) [17], while yielding much higher sample efficiency. Since our primary interest is MPC and its application to practical systems, this paper mainly focuses on PETS and treats this method as a strong baseline.

Considering the success of probabilistic dynamics modeling, incorporating uncertainty in optimal trajectories appears very promising for MBRL. However, an optimization scheme that can utilize uncertainty has not yet been discussed. Although several stochastic approaches, including CEM, model predictive path integral (MPPI) [18, 8], covariance matrix adaptation evolution strategy (CMA-ES) [19], and proportional CEM (Prop-CEM) [20], have been proposed, they are not uncertainty-aware and tend to underestimate uncertainty. In addition, although their optimization procedures are very similar, they have been independently derived. Consequently, theoretical relations among these methods are unclear, preventing us from systematically understanding and reformulating them to be uncertainty-aware in a Bayesian fashion.

Motivated by these, in this paper, we propose a novel MPC concept for Bayesian MBRL. The organization and contributions of this paper are summarized as follows. (1) In Sec. 3, we introduce a novel MPC framework, variational inference MPC (VI-MPC), which generalizes and reformulates various stochastic MPC methods in a Bayesian fashion. The key observations for deriving this framework are organized in Sec. 2, where we point out that general stochastic optimization methods can be regarded as the moment matching of the optimal trajectory posterior, which appear in a Bayesian MBRL formulation. (2) In Sec. 4, we propose a novel instance of the framework, called probabilistic action ensembles with trajectory sampling (PaETS). Toy task examples and the concept of our method are exhibited in Fig. 1. (3) In Sec. 5, we demonstrate that our method consistently outperforms PETS via experiments with challenging locomotion tasks in the MuJoCo physics simulator [21].

2 Model-based Reinforcement Learning as Bayesian Inference

In this section, we describe MBRL as a Bayesian inference problem using *control as inference* framework [22]. Fig. 2 displays the graphical model for the formulation, with which an MBRL procedure can be re-written in a Bayesian fashion: (1. *training-step*) do inference of $p(\theta|\mathcal{D})$. (2. *test-step*) do inference of $p(\tau|\mathcal{O}_{1:T} = \mathbf{1})$, then, sample actions from the posterior and execute the actions in a real environment. We denote a trajectory as $\tau := \{(\boldsymbol{s}_t, \boldsymbol{a}_t)\}_{t=1}^{T}$, where $\boldsymbol{s}_t$ and $\boldsymbol{a}_t$ respectively represent state and action. Given a state-action pair at time t, the next state can be predicted by a forward-dynamics model $\boldsymbol{s}_{t+1} \sim p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t, \theta)$ parameterized with θ. The posterior of θ is inferred from training dataset $\mathcal{D}$, where $\mathcal{D} = \{(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})\}$ consists of states and actions *observed* during the test step. To formulate optimal control as inference, we auxiliarly introduce a binary random variable $\mathcal{O}_t \in \{0, 1\}$ to represent the *optimality* of $(\boldsymbol{s}_t,\ \boldsymbol{a}_t)$. Given $p(\theta|\mathcal{D})$, trajectory optimization can be expressed as an inference problem:

$$p(\tau|\mathcal{O}) \propto \int \underbrace{\left\{\prod_{t=1}^{T} p(\mathcal{O}_t = 1|\boldsymbol{s}_t, \boldsymbol{a}_t)\right\}}_{:=p(\mathcal{O}|\tau)} \cdot \underbrace{p(\boldsymbol{s}_1)\left\{\prod_{t=1}^{T} p\left(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t, \theta\right)\right\}}_{:=p(\boldsymbol{s}|\boldsymbol{a},\theta)} \cdot \underbrace{p(\theta|\mathcal{D})}_{:=p_{\mathcal{D}}(\theta)} d\theta, \qquad (1)$$

where uninformative action prior (i.e., $p(\boldsymbol{a}_t) = \mathcal{U}$: uniform distribution) is supposed. For readability, $\mathcal{O}_{1:T} = \mathbf{1}$ is simply denoted as $\mathcal{O}$. For the same reason, we omit the subscripts of sequences $\boldsymbol{a}_{1:T}$, $\boldsymbol{s}_{1:T}$. In the remainder of the paper, this simplified notation is employed. In Sec. 2.1–2.2, we review how these inference problems have been approximately handled in previous works.

2.1 Inference of Forward-dynamics Posterior $p_{\mathcal{D}}(\theta)$

Given a sufficiently parameterized expressive model, i.e., DNNs, one of the most practical and promising schemes for approximating the posterior $p_{\mathcal{D}}(\theta)$ is to utilize neural network ensembles [13, 14, 15]. This process approximates the posterior as a set of *particles* $p_{\mathcal{D}}(\theta) \simeq \frac{1}{E}\sum_{i}^{E} \delta(\theta - \theta_i)$, where δ is Dirac delta function and E is the number of networks. Each particle θ_i is independently trained by stochastic gradient descent so as to (sub-)optimize $\log p_{\mathcal{D}}(\theta) \propto \log p(\mathcal{D}|\theta)p(\theta)$. Although this approximation is incompletely Bayesian, this scheme has several useful features. First, we can simply implement this process in standard deep learning frameworks. Furthermore, the ensemble model successfully involves multimodal uncertainty in the exact posterior.

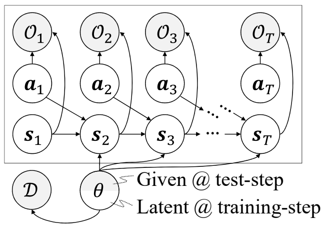


Figure 2: Graphical model for Bayesian MBRL.

Another possible way to infer $p_{\mathcal{D}}(\theta)$ is *dropout as variational inference* [10, 11, 12], which approximates $p_{\mathcal{D}}(\theta)$ as a Gaussian distribution $q(\theta)$. It is proofed that the variational inference problem: $\mathrm{argmin}_q\, \mathrm{KL}(q(\theta)||p_{\mathcal{D}}(\theta))$ approximately equivalent to training networks with dropout, where $\mathrm{KL}(\cdot||\cdot)$ denote Kullback-Leibler (KL) divergence. Although this scheme is also simple and theoretically supported, approximation by a single Gaussian distribution tends to underestimate uncertainty (or multimodality) in the posterior. To remedy this problem, α-divergence dropout has been proposed [23], which replaces KL-divergence to α-divergence so as to prevent $q(\theta)$ from overfitting a single mode. However, as long as $q(\theta)$ is Gaussian, the multimodality cannot be managed well.

In our preliminary experiments of MBRL, we have tested the above two schemes and observed that the ensemble performs much better than (α-)dropout (this result is summarized in Sec. A). This result provides us with the insight that capturing multimodality in the posterior has crucial effects in MBRL literature. Therefore, in this paper, we also employ this ensemble scheme to approximate $p_{\mathcal{D}}(\theta)$ in the same way as our baseline: PETS [13]. In Sec. 4, we also attempt to incorporate multimodality in the posterior $p(\tau|\mathcal{O})$.

2.2 Moment Matching of Trajectory Posterior $p(\tau|\mathcal{O})$

This section clarifies the connection between trajectory optimization and the posterior approximation problem. The key observation delineated here is that several MPC methods, including CEM used in PETS and MPPI, can be regarded as the moment matching of the posterior.

Table 1: Optimization algorithms derived by moment matching of $p(\tau|\mathcal{O})$ and different f definitions; $\mathbb{1}$ indicates an indicator function, $g : \mathbb{R} \mapsto \mathbb{N}$ denotes rank-preserving transformation.

	MPPI [18]	CEM [16]	Prop-CEM [20]	CMA-ES [19]
$f(r(\tau))$	$\propto e^{r(\tau)}$	$\mathbb{1}\left[r(\tau) > r_{thd}\right]$	$\dfrac{r(\tau) - r_{min}}{r_{max} - r_{min}}$	$\propto \log g(r(\tau)) \cdot \mathbb{1}\left[r(\tau) > r_{thd}\right]$

Given an inferred model posterior $p_{\mathcal{D}}(\theta)$, we can sample trajectories from (1).[1] Let us approximate the action posterior with a Gaussian distribution $q(\boldsymbol{a}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$. The mean of posterior action sequence $\boldsymbol{\mu}$ can be estimated by moment matching:

$$\boldsymbol{\mu} = \mathbb{E}\left[\boldsymbol{a} \cdot p(\tau|\mathcal{O})\right] = \frac{\mathbb{E}_{\boldsymbol{s}\sim p(\boldsymbol{s}|\boldsymbol{a},\theta),\theta\sim p_{\mathcal{D}}(\theta),\boldsymbol{a}\sim\mathcal{U}}\left[\boldsymbol{a} \cdot p(\mathcal{O}|\tau)\right]}{\mathbb{E}_{\boldsymbol{s}\sim p(\boldsymbol{s}|\boldsymbol{a},\theta),\theta\sim p_{\mathcal{D}}(\theta),\boldsymbol{a}\sim\mathcal{U}}\left[p(\mathcal{O}|\tau)\right]} = \frac{\mathbb{E}_{\boldsymbol{a}\sim\mathcal{U}}\left[\boldsymbol{a} \cdot \mathcal{W}(\boldsymbol{a})\right]}{\mathbb{E}_{\boldsymbol{a}\sim\mathcal{U}}\left[\mathcal{W}(\boldsymbol{a})\right]}, \tag{2}$$

where

$$\mathcal{W}(\boldsymbol{a}) := \mathbb{E}_{\boldsymbol{s}_{t+1}\sim p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t,\boldsymbol{a}_t,\theta),\theta\sim p_{\mathcal{D}}(\theta)}\left[p(\mathcal{O}|\tau)\right]. \tag{3}$$

Eq. (2) can be viewed as a weighted average where each sampled action is weighted by the likelihood of optimality $\mathcal{W}(\boldsymbol{a})$. In the same way, we can also estimate the variance of the posterior $\boldsymbol{\Sigma} = \mathbb{E}_{\boldsymbol{a}\sim\mathcal{U}}\left[(\boldsymbol{a} - \boldsymbol{\mu})^2\mathcal{W}(\boldsymbol{a})\right]/\mathbb{E}_{\boldsymbol{a}\sim\mathcal{U}}\left[\mathcal{W}(\boldsymbol{a})\right]$.

In practice, sampling from uniform distribution $\mathcal{U}$ is quite inefficient and requires almost infinite samples. Hence, let us consider iteratively estimating the parameters by incorporating importance sampling. Let $\boldsymbol{\mu}^{(j)}$, $\boldsymbol{\Sigma}^{(j)}$ be the estimated parameters at iteration j; we can rearrange (2) as

$$\boldsymbol{\mu}^{(j+1)} \leftarrow \{\text{RHS of (2)}\} \times \frac{q(\boldsymbol{a}; \boldsymbol{\mu}^{(j)}, \boldsymbol{\Sigma}^{(j)})}{q(\boldsymbol{a}; \boldsymbol{\mu}^{(j)}, \boldsymbol{\Sigma}^{(j)})} = \frac{\mathbb{E}_{\boldsymbol{a}\sim q(\boldsymbol{a};\boldsymbol{\mu}^{(j)},\boldsymbol{\Sigma}^{(j)})}\left[\boldsymbol{a} \cdot \mathcal{W}(\boldsymbol{a})\right]}{\mathbb{E}_{\boldsymbol{a}\sim q(\boldsymbol{a};\boldsymbol{\mu}^{(j)},\boldsymbol{\Sigma}^{(j)})}\left[\mathcal{W}(\boldsymbol{a})\right]}. \tag{4}$$

It is worth noting that a similar iterative law can also be derived by solving the optimization problem $\operatorname{argmax}_{q(\boldsymbol{a};\boldsymbol{\mu},\boldsymbol{\Sigma})} \mathbb{E}\left[\log p(\mathcal{O}|\tau)\right]$ by mirror descent [24, 25]. To connect this inference problem to trajectory optimization, we define the optimality likelihood with trajectory reward $r(\tau)$ and a monotonically increasing function $f(\cdot)$, as $p(\mathcal{O}|\tau) := f(r(\tau))$. If we define $f(r(\tau)) \propto e^{r(\tau)}$ the same as [22, 26], an optimization algorithm similar to MPPI [18, 8, 25] is recovered. As summarized in Table 1, other similarities to well-known optimization algorithms, including CEM, can be observed with different optimality definitions. [2]

There is a discrepancy between (4) and the CEM implementation in [13]; in which $\mathcal{W}'(\boldsymbol{a}) = f(\mathbb{E}[r(\tau)])$ is used instead of $\mathcal{W}(\boldsymbol{a}) = \mathbb{E}[f(r(\tau))]$. Since f is a convex function, Jensen's inequality holds in this case, thus $\mathcal{W} \geq \mathcal{W}'$. The equality holds when $f(\cdot)$ is constant, implying that $\mathcal{W} \simeq \mathcal{W}'$ for low-variance $r(\tau)$ and $\mathcal{W} > \mathcal{W}'$ for high-variance (or more uncertain) $r(\tau)$. Namely, $\mathcal{W}'(\boldsymbol{a})$ underestimates the optimality likelihood if $\boldsymbol{a}$ generates uncertain trajectories. Since we have experimentally observed that this uncertainty avoidance behavior by $\mathcal{W}'$ demonstrates higher optimization performance than $\mathcal{W}$ (see Sec. B), this paper heuristically employs the use of $\mathcal{W}'$.

In practice, expectation operators $\mathbb{E}[\cdot]$ should be implemented on digital computers through the Monte Carlo integration with K sampled actions and P trajectories for each action: $\boldsymbol{\mu}^{(j+1)} \simeq \sum_{k=1}^{K}\left[\boldsymbol{a}_k \cdot \mathcal{W}'(\boldsymbol{a}_k)\right]/\sum_{k=1}^{K}\left[\mathcal{W}'(\boldsymbol{a}_k)\right]$ and $\mathcal{W}'(\boldsymbol{a}_k) \simeq f\left(\frac{1}{P}\sum_{i=1}^{P} r(\tau_{k,i})\right)$.

3 Variational Inference MPC: From Moment Matching to Inference

Given uncertainty in a dynamics model, it is natural to suppose that optimal trajectories are also uncertain. However, as exhibited in the previous section, PETS employs the moment matching of the trajectory posterior, ignoring almost uncertainty in optimal trajectories. In this section, we newly introduce a variational inference MPC (VI-MPC) framework to formulate MBRL as fully Bayesian and involve uncertainty both in the dynamics and optimalities.

[1] Trajectory sampling methods with $p_{\mathcal{D}}(\theta)$ have been discussed and experimented in [13]. In this paper, we employ the TS1 method suggested in the reference (see ℓ3–6 in Alg. 1).

[2] We implicitly assume the existence of step-wise likelihood $p(\mathcal{O}_t|\boldsymbol{s}_t, \boldsymbol{a}_t)$ corresponding to each definition. Since another graphical model with a single unified optimality can be defined, the existence is not critical.

Let us consider a variational inference problem: $\mathrm{KL}\left(q_\theta(\tau)||p(\tau,\theta|\mathcal{O})\right)$. We assume the variational distribution $q_\theta(\tau)$ is decomposed to $q_\theta(\tau) = q(\boldsymbol{a})p(\boldsymbol{s}|\boldsymbol{a},\theta)p_\mathcal{D}(\theta)$; hence, we introduce $p(\tau,\theta|\mathcal{O})(= p(\mathcal{O}|\tau)p(\boldsymbol{s}|\boldsymbol{a},\theta)p_\mathcal{D}(\theta))$ as a posterior, which takes the similar decomposable form as $q_\theta(\tau)$. This assumption forces optimal state transitions to be controlled only by $p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t,\boldsymbol{a}_t,\theta)$ [22]. As shown in Sec. C.1, this inference problem can be transformed to the maximization problem: $\mathrm{argmax}_{q_\theta(\tau)}\,\mathbb{E}\left[\log p(\mathcal{O}|\tau) - \log q(\boldsymbol{a})\right]$. A notable property is that this objective has an entropy regularization term $-\log q(\boldsymbol{a})$, which encourages $q(\boldsymbol{a})$ to have broader shape to capture more uncertainty. For the sake of convenience, we introduce a tunable hyperparameter $\alpha(>0)$ to the optimality likelihood $p(\mathcal{O}|\tau) \to p^{\frac{1}{\alpha}}(\mathcal{O}|\tau)$. Then the above objective can be transformed as $\mathrm{argmax}_{q_\theta(\tau)}\,\mathbb{E}\left[\log p(\mathcal{O}|\tau) - \alpha\log q(\boldsymbol{a})\right]$. By applying mirror descent [27] to this optimization problem, we can derive an update law for $q(\boldsymbol{a})$ (see Sec. C.2 for the detailed derivation):

$$q^{(j+1)}(\boldsymbol{a}) \leftarrow q^{(j)}(\boldsymbol{a}) \cdot \mathcal{W}'(\boldsymbol{a})^{\frac{1}{\lambda}} \cdot (q^{(j)}(\boldsymbol{a}))^{-\kappa} \Big/ \mathbb{E}_{q^{(j)}(\boldsymbol{a})}\left[\mathcal{W}'(\boldsymbol{a})^{\frac{1}{\lambda}} \cdot (q^{(j)}(\boldsymbol{a}))^{-\kappa}\right], \tag{5}$$

where $\lambda(>0)$, $\kappa(>0)$ are hyperparameters and α is absorbed into them. λ is inverted step-size to control optimization speed and κ is the weight of the entropy regularization term $q^{-\kappa}$.

Eq. (5) suggests a novel and general MPC framework, which we call variational inference MPC (VI-MPC). To realize a specific VI-MPC method, we specify the following parameters: (1) optimality definition (or $f(\cdot)$; see Table 1), (2) variational distribution model q, and (3) entropy regularization $\kappa > 0$ or $\kappa = 0$. We did not include λ into the specifications since it is highly dependent on the optimality definition (see Sec. G). In this paper, we describe the above specifications as `VIMPC(<optimality_def>, <variational_dist>, <max_ent>)`. For example, we respectively express vanilla CEM and MPPI as `VIMPC('CEM', 'Gaussian', False)` and `VIMPC('MPPI', 'Gaussian', False)`. In Sec. 4, we propose a new instance of VI-MPC to incorporate multimodal uncertainty in the posterior.

4 Probabilistic Action Ensembles with Trajectory Sampling

As reviewed in Sec. 2.1, previous methods have successfully involved multimodality in $p_\mathcal{D}(\theta)$ with network ensembles. If this multimodality in $p_\mathcal{D}(\theta)$ is given, other distributions depending on $p_\mathcal{D}(\theta)$, including $p(\mathcal{O}|\tau)$, would also be multimodal. In other words, there are various possible optimal trajectories (or actions) like Fig. 1. It is obvious that `VIMPC(*, 'Gaussian', *)` will still easily fail to capture multimodality because of overfitting to a single mode. Inspired by the success of the ensemble approach for dynamics modeling, we propose a novel VI-MPC method that introduces action ensembles with a Gaussian mixture model (GMM), i.e., `VIMPC(*, 'GMM(M=*)', *)`, which we call PaETS (Probabilistic Action Ensembles with Trajectory Sampling).

PaETS defines the variational distribution $q(\boldsymbol{a})$ as

$$q^{(j)}(\boldsymbol{a}) := q(\boldsymbol{a};\phi^{(j)}) = \sum_{m=1}^{M} \pi_m^{(j)} \mathcal{N}(\boldsymbol{a};\boldsymbol{\mu}_m^{(j)},\boldsymbol{\Sigma}_m^{(j)}), \tag{6}$$

where $\phi^{(j)} := \{(\pi_m^{(j)},\boldsymbol{\mu}_m^{(j)},\boldsymbol{\Sigma}_m^{(j)})\}_{m=1}^{M}$ and M is the number of components of the mixture model. Now, we derive the iteration scheme to update the parameters of GMM. At first, drawing K samples from $q^{(j)}(\boldsymbol{a})$, we approximate $q^{(j)}(\boldsymbol{a})$ as a discretized distribution (or a set of particles):

$$q^{(j)}(\boldsymbol{a};\phi) \simeq q(\boldsymbol{a};\mathbf{W}^{(j)}) := \sum_{k=1}^{K} w_k^{(j)} \delta(\boldsymbol{a} - \boldsymbol{a}_k), \tag{7}$$

where $\mathbf{W}^{(j)} := \{w_k^{(j)}\}_{k=1}^{K}$. Just after sampling, the weight of each particle is uniform: $\mathbf{W}^{(j)} = \mathbf{1}/K$. By substituting this approximated distribution to (5), the update law for the particle weights is derived as

$$w_k^{(j+1)} \leftarrow \mathcal{W}'(\boldsymbol{a}_k)^{\frac{1}{\lambda}} \cdot (q^{(j)}(\boldsymbol{a}_k))^{-\kappa} \Big/ \sum_{k'=1}^{K} \mathcal{W}'(\boldsymbol{a}_{k'})^{\frac{1}{\lambda}} \cdot (q^{(j)}(\boldsymbol{a}_{k'}))^{-\kappa}. \tag{8}$$

Then we estimate $\phi^{(j+1)}$, which maximizes the observation probability of the weighted particles:

$$\phi^{(j+1)} = \mathrm{argmax}_\phi \log p(\{(w_k^{(j+1)},\boldsymbol{a}_k)\}_{k=1}^{K}|\phi) = \mathrm{argmax}_\phi \sum_{k=1}^{K} w_k^{(j+1)} \log q(\boldsymbol{a}_k;\phi). \tag{9}$$

Algorithm 1: PaETS

```
Input: State $s_1$, GMM param. $\phi^{(1)}$ and $p_{\mathcal{D}}(\theta)$
Output: Optimized GMM param. $\phi^{(U+1)}$
1 for $j \leftarrow 1$ to $U$ do
2     Sample actions $\{a_k \sim q(a;\phi^{(j)})\}_{k=1}^{K}$
3     Sample states $\{\{\{$
4         $\theta_{k,i,t} \sim p_{\mathcal{D}}(\theta)$ // TS1 method
5         $s_{k,i,t+1} \sim$
              $p(s_{t+1}|a_{k,t}, s_{k,i,t}, \theta_{k,i,t})$,
6     $\}_{t=1}^{T-1}\}_{i=1}^{P}\}_{k=1}^{K}$
7     Eval. $\{\mathcal{W}'(a_k) \simeq f(\sum_{i=1}^{P} r(\tau_{k,i}))\}_{k=1}^{K}$
8     Calc. $\{w_k^{(j+1)}\}_{k=1}^{K}$ by (8)
9     Update $\phi^{(j+1)}$ by (10)
```

Algorithm 2: MBRL with PaETS

```
Data: initial variance $\Sigma_{init}$
1 Init. $\mathcal{D}$ with a random controller for one trial
2 repeat
3     Infer $p_{\mathcal{D}}(\theta)$ // train ensemble DNNs
4     $\{\mu_m \leftarrow \mathcal{N}(a; 0, \Sigma_{init})\}_{m=1}^{M}$ // rand. init.
5     $\{(\Sigma_m, \pi_m) \leftarrow (\Sigma_{init}, 1/M)\}_{m=1}^{M}$
6     for $n \leftarrow 1$ to $H$ do
7         $\phi \leftarrow$Exec. Alg. 1$(s_n, \phi, p_{\mathcal{D}}(\theta))$
8         Sample $a \sim q(a;\phi)$
9         Send $a_1$ to actuators and observe $s_{n+1}$
10        $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_n, a_1, s_{n+1})\}$
11        $\{\mu_m \leftarrow \{\mu_{m,2:T}, 0\}\}_{m=1}^{M}$ // warm-start
12        $\{(\Sigma_m, \pi_m) \leftarrow (\Sigma_{init}, 1/M)\}_{m=1}^{M}$
13 until the MPC-policy performs well
```

By taking the derivative $\nabla_\phi \log p(\cdot|\phi) = \mathbf{0}$ and borrowing the concept of the EM algorithm [28], we get the update laws of $\phi^{(j+1)}$ which take the weight-average form like (4) (see Sec. D for the complete definition):

$$\left(\boldsymbol{\mu}_m^{(j+1)}, \boldsymbol{\Sigma}_m^{(j+1)}, \pi_m^{(j+1)}\right) \leftarrow \left(\sum_{k=1}^{K} \omega_{m,k}^{(j+1)} \boldsymbol{a}_k, \sum_{k=1}^{K} \omega_{m,k}^{(j+1)} (\boldsymbol{a}_k - \boldsymbol{\mu}_m^{(j+1)})^2, \frac{N_m}{\sum_{m=1}^{M} N_m}\right). \quad (10)$$

Fig. 8 in Sec. E illustrates how this method works in a toy optimization task.

In summary, PaETS and MBRL utilizing it are respectively described in Algs. 1 and 2, where U is the number of iterations for optimization and H is the length of the task episode. At $\ell 4$ in Alg. 2, $\boldsymbol{\mu}_m$s are initialized independently at random. At $\ell 12$, $\boldsymbol{\Sigma}_m$s and π_ms are reset to be initial values, encouraging exploration for the next timestep and preventing $q(\boldsymbol{a};\phi)$ from degenerating to a single mode. If we set $M = 1$, these procedures are almost equivalent to those of PETS. The use of GMM ($M > 1$) does not increase computational complexity significantly (see Sec. F).

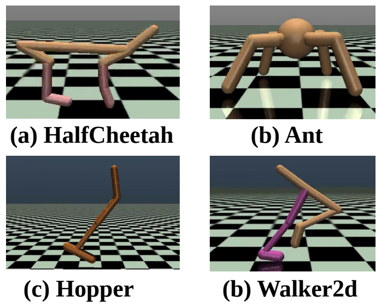


Figure 3: Evaluated locomotion tasks simulated in MuJoCo.

5 Experiments

5.1 Comparison to State-of-the-art Methods

The main objective of this experiment is to demonstrate that PaETS has advantages over the state-of-the-art MBRL baseline: PETS [13]. In this experiment, PaETS and PETS (or vanilla CEM) were implemented using our same codebase with different parameters, i.e., `VIMPC('CEM', 'GMM(M=5)', True)` for PaETS, and `VIMPC('CEM', 'GMM(M=1)', False)` for PETS. We also evaluated another MBRL baseline with MPPI [8], realized as `VIMPC('MPPI', 'GMM(M=1)', False)`. These above methods share the settings for $p_{\mathcal{D}}(\theta)$ inference (training of network ensembles). The state-of-the-art MFRL method SAC [17], was also evaluated to compare asymptotic performance.[3] Fig. 3 illustrates the simulated locomotion tasks evaluated in this experiment, which are complex and challenging due to their high non-linearity. Other details about our implementation and experimental settings are described in Sec. G and Sec. H. Fig. 4 presents the experimental results, in which PaETS consistently exhibits better asymptotic performance than that of the MBRL baselines. In addition, PaETS outperforms or is comparable to SAC while requiring significantly fewer samples (about x10 more sample efficient).

[3]We used the open-source code: `https://github.com/pranz24/pytorch-soft-actor-critic`

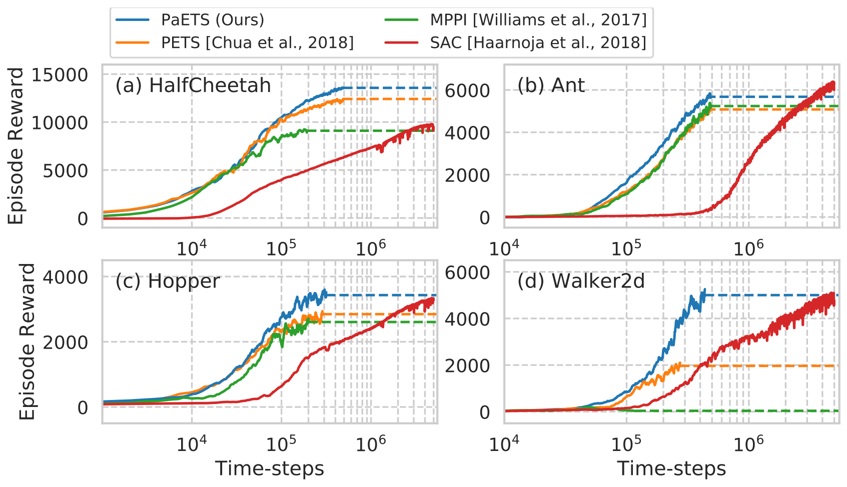


Figure 4: Learning curves for different tasks and algorithms. These are averaged results of 8 (for MBRL) and 20 (for SAC) trials with different random seeds. We stopped the training when convergence was observed or after reaching the specified test steps (500 for MBRL and $5,000$ for SAC). The asymptotic performances (averages of the last 10 test steps) are depicted in dashed lines.

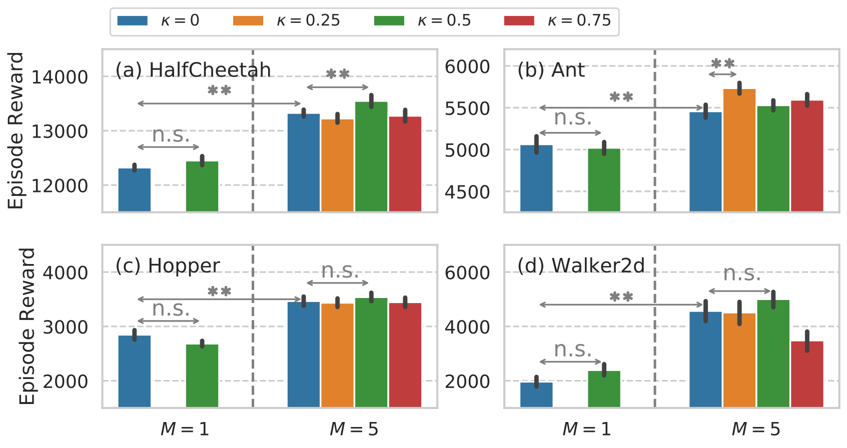


Figure 5: Asymptotic performance comparison with varying Ms and κs. These are averaged results over 8 different MBRL trials and the last 10 test steps. The error bars denote confidence intervals (95%). Symbols '**' and 'n.s.' respectively mean $p < 0.01$ and $p \geq 0.01$ in Welch's t-test.

5.2 Ablation Study

This experiment clarifies which component of PaETS (GMM and entropy-regularization) contributed to the overall improvement. Fig. 5 expresses the results of this ablation study and Welch's t-test for some selected representative pairs. From this figure, one can observe that the use of GMM ($M = 5$) significantly improves performance. The effect of the regularization ($\kappa > 0$) is relatively small, but not negligible. In certain tasks, setting κ to particular values could improve the performance. In the case of $M > 1$, the regularization sheds light on actions sampled from low π_m, thus encouraging $q(\boldsymbol{a}; \phi)$ to be multimodal. In some tasks which requires rather delicate controls (e.g., Hopper, Walker2d), the effect of κ seems less significant. Fig. 6 examines sensitivity with the number of mixture components M, for which $M = 5$ achieved the highest performance. If infinite or enough samples are given ($K \gg 0$), it would be reasonable to set M to be large enough to capture multimodality. However, in practice, K is finite and could be small enough due to computational constraints. In this case, larger M makes it difficult to approximate $q(\boldsymbol{a}; \phi)$ as a set of particles $q(\boldsymbol{a}; \mathbf{W})$, resulting in degradation of the optimization performance.

6 Related Work

Dynamics Posterior Inference Recent MBRL methods, MB-MPO (Model-Based Meta-Policy-Optimization) [15] and ME-TRPO (Model Ensemble Trust Region Optimization) [14], also employ network ensembles to model dynamics, but they utilize the ensembles differently than we do: to train policy networks, not MPC.

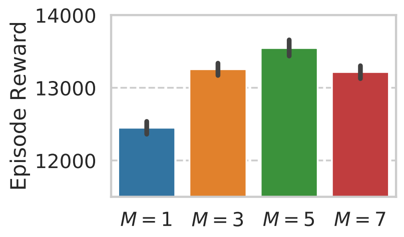


Figure 6: Asymptotic performance comparison with varying Ms. Only the HalfCheetah task is evaluated in this test.

Trajectory Optimization Sequential Monte-Carlo based MPC, described as `VIMPC(*, 'Particles', False)`, has been introduced in [29], but it requires well-designed proposal distribution to sample particles for the next iteration $j + 1$. Another particle-based method has been derived [26] by utilizing the *control as inference* framework. However, this method relies on not only a dynamics model, but also policy and value functions to manage particles, so MFRL methods must be incorporated.

Recent studies have demonstrated that entropy regularization is a promising strategy in policy training [30, 31, 32, 17]. However, to the best of our knowledge, the introduction of entropy regularization to MPC is novel along with explicit multimodal expression to successfully realize their synergistic effect.

Ref. [33] also systematically organizes the stochastic MPC methods from the perspective of online learning, but uncertainty-aware discussions from a Bayesian viewpoint are not conducted.

Bayesian Reformulation Ref. [34] proposes a novel approach to generative adversarial imitation learning (GAIL) [35], which reformulates general GAIL in a Bayesian fashion and utilizes ensembles to infer discriminator posteriors. Another Bayesian reformulation of GAIL integrates imitation and reinforcement learning by introducing another optimality (i.e., imitation optimality $\mathcal{O}_t^I$) [36].

7 Conclusion & Discussions

This paper introduces a novel VI-MPC framework that systematically generalizes and reformulates various stochastic MPC methods in a Bayesian fashion. We also devise a novel instance of this framework, called PaETS, which can successfully incorporate multimodal uncertainty in optimal trajectories. By combining our method and the recent uncertainty-aware dynamics modeling with neural network ensembles, our Bayesian MBRL is able to involve multimodalities both in dynamics and optimalities. In addition, our method is a quite simple extension of general stochastic methods and requires no significant additional computational complexity. Our experiments demonstrate that PaETS can improve asymptotic performance compared to the leading MBRL baseline PETS, and thus substantially enhances MBRL potential to be more competitive to the state-of-the-art MFRL.

Considering the simplicity and generalizability of VI-MPC and PaETS, we expect that our concept is applicable to a variety of tasks, such as traditional MPC with deterministic dynamics and advanced MPC with latent dynamics from pixels by Deep Planning Network [37]. By introducing a categorical mixture model as a variational distribution, application to combinational optimizations is also feasible. In fact, our ongoing work includes experiments of discrete MPC for a practical system.

A question that remains is how to determine VI-MPC specifications. As implied in Fig. 4, the best optimality definition could be task dependent (e.g., MPPI outperformed vanilla CEM in the Ant but not in other tasks). The regularization weight κ also has task dependency as shown in Fig. 5. It would be challenging but interesting future work to add the parameters to the graphical model in Fig. 2 as latent variables to infer promising parameters along with optimal trajectories, like infinite GMM [38]. Another appealing endeavor for future work is to introduce the concept of parallel tempering [39] in Markov Chain Monte Carlo. By adaptively varying different temperatures (λ in our case) of ensemble actions, we can expect the ensemble diversity to improve.

Acknowledgments

We thank Vishwajeet Singh, Hiroki Nakamura and Akira Kinose for their cooperation in this study during their student-internship periods. Most of the experiments were conducted in ABCI (*AI Bridging Cloud Infrastructure*), built by the National Institute of Advanced Industrial Science and Technology, Japan.

References

[1] F. D. Vargas-Villamil and D. E. Rivera. Multilayer optimization and scheduling using model predictive control: application to reentrant semiconductor manufacturing lines. *Computers & Chemical Engineering*, 24(8):2009–2021, 2000.

[2] A. Afram and F. Janabi-Sharifi. Theory and applications of HVAC control systems–a review of model predictive control (MPC). *Building and Environment*, 72:343–355, 2014.

[3] S. Vazquez, J. Leon, L. Franquelo, J. Rodriguez, H. A. Young, A. Marquez, and P. Zanchetta. Model predictive control: A review of its applications in power electronics. *IEEE Ind. Electron. Mag.*, 8(1):16–31, 2014.

[4] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Trans. Intell. Veh.*, 1(1):33–55, 2016.

[5] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake. Optimization-based locomotion planning, estimation, and control design for the Atlas humanoid robot. *Autonomous Robots*, 40(3):429–455, 2016.

[6] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

[7] M. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *ICML*, 2011.

[8] 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 *ICRA*, 2017.

[9] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *ICRA*, 2018.

[10] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016.

[11] Y. Gal, J. Hron, and A. Kendall. Concrete dropout. In *NeurIPS*, 2017.

[12] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine. Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint arXiv:1702.01182*, 2017.

[13] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, 2018.

[14] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel. Model-ensemble trust-region policy optimization. In *ICLR*, 2018.

[15] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel. Model-based reinforcement learning via meta-policy optimization. In *CoRL*, 2018.

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

[17] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.

[18] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou. Aggressive driving with model predictive path integral control. In *ICRA*, 2016.

[19] N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary computation*, 11(1):1–18, 2003.

[20] S. Goschin, A. Weinstein, and M. Littman. The cross-entropy method optimizes for quantiles. In *ICML*, 2013.

[21] E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. In *IROS*, 2012.

[22] S. Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

[23] Y. Li and Y. Gal. Dropout inference in bayesian neural networks with alpha-divergences. In *ICML*, 2017.

[24] M. Miyashita, S. Yano, and T. Kondo. Mirror descent search and its acceleration. *Robotics and Autonomous Systems*, 106:107–116, 2018.

[25] M. Okada and T. Taniguchi. Acceleration of gradient-based path integral method for efficient optimal and inverse optimal control. In *ICRA*, 2018.

[26] A. Piche, V. Thomas, C. Ibrahim, Y. Bengio, and C. Pal. Probabilistic planning with sequential monte carlo methods. In *ICLR*, 2018.

[27] S. Bubeck et al. *Convex optimization: Algorithms and complexity*, volume 8, chapter 4. Now Publishers, Inc., 2015.

[28] J. A. Bilmes et al. A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute*, 4(510):126, 1998.

[29] N. Kantas, J. Maciejowski, and A. Lecchini-Visintini. Sequential monte carlo for model predictive control. In *Nonlinear model predictive control*, pages 263–273. Springer, 2009.

[30] A. Abdolmaleki, R. Lioutikov, J. R. Peters, N. Lau, L. P. Reis, and G. Neumann. Model-based relative entropy stochastic search. In *NeurIPS*, 2015.

[31] A. Abdolmaleki, B. Price, N. Lau, L. P. Reis, and G. Neumann. Deriving and improving CMA-ES with information geometric trust regions. In *GECCO*, 2017.

[32] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. In *ICML*, 2017.

[33] N. Wagener, C.-A. Cheng, J. Sacks, and B. Boots. An online learning approach to model predictive control. In *Robotics: Science and Systems*, 2019.

[34] W. Jeon, S. Seo, and K.-E. Kim. A bayesian approach to generative adversarial imitation learning. In *NeurIPS*, 2018.

[35] J. Ho and S. Ermon. Generative adversarial imitation learning. In *NeurIPS*, 2016.

[36] A. Kinose and T. Tadahiro. Integration of imitation learning using GAIL and reinforcement learning using task-achievement rewards via probabilistic generative model. *arXiv preprint arXiv:1907.02140*, 2019.

[37] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *ICML*, 2019.

[38] C. E. Rasmussen. The infinite gaussian mixture model. In *NeurIPS*, 2000.

[39] S. Brooks, A. Gelman, G. Jones, and X.-L. Meng. *Handbook of Markov Chain Monte Carlo*, chapter 11. CRC press, 2011.

[40] E. Theodorou, J. Buchli, and S. Schaal. A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research*, 11:3137–3181, 2010.

A Preliminary Experiment for Uncertainty Modeling

Fig. 7 shows the result of a preliminary experiment, in which different uncertainty modeling approaches were evaluated on the HalfCheetah task. For all trials, vanilla CEM was introduced for trajectory optimization. This result suggests that (α-)dropout is insufficient to capture uncertainty in dynamics, resulting in worse local optima.

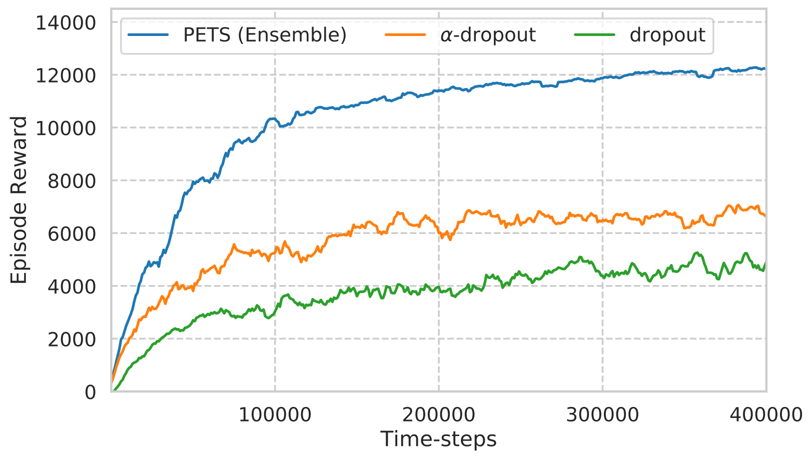


Figure 7: Comparison of uncertainty modeling approaches: ensemble and (α-)dropout.

B Comparison Between $\mathcal{W}$ and $\mathcal{W}'$

We evaluated the impact of $\mathcal{W}$ and $\mathcal{W}'$ on the optimization performance of (vanilla) CEM and MPPI, the results of which are summarized in Table 2, where $\mathcal{W}'$ gained much higher rewards than $\mathcal{W}$.

Table 2: Episode reward of HalfCheetah task with $\mathcal{W}$ and $\mathcal{W}'$. A common dynamics model (sufficiently trained ensemble neural network by MBRL) was employed for this test. Ten different trials were conducted and the results were averaged.

CEM		MPPI	
$\mathcal{W}$	$\mathcal{W}'$	$\mathcal{W}$	$\mathcal{W}'$
5603.24 ± 541.31	$\mathbf{11843.05 \pm 295.80}$	2789.03 ± 647.82	$\mathbf{9765.27 \pm 231.04}$

C Derivations

C.1 Derivation of the Variational Inference Objective

By using the assumption of $q_\theta(\tau) = q(\boldsymbol{a})p(\boldsymbol{s}|\boldsymbol{a},\theta)p_{\mathcal{D}}(\theta)$, the KL-divergence can be transformed as

$$\mathrm{KL}\left(q_\theta(\tau)||p(\tau,\theta|\mathcal{O})\right) = \int q_\theta(\tau)\log\frac{q_\theta(\tau)}{p(\tau,\theta|\mathcal{O})}d\tau d\theta \tag{11}$$

$$= \int q_\theta(\tau)\log\frac{q(\boldsymbol{a})p(\boldsymbol{s}|\boldsymbol{a},\theta)p_{\mathcal{D}}(\theta)}{p(\mathcal{O}|\tau)p(\boldsymbol{s}|\boldsymbol{a},\theta)p_{\mathcal{D}}(\theta)}d\tau d\theta \tag{12}$$

$$= -\mathbb{E}_{q_\theta(\tau)}\left[\log p(\mathcal{O}|\tau) - \log q(\boldsymbol{a})\right]. \tag{13}$$

C.2 Derivation of (5)

In this section, we simply denote $q_{\boldsymbol{a}}$ as $q(\boldsymbol{a})$ and q_τ as $q(\tau)(= q_{\boldsymbol{a}} p(\boldsymbol{s}|\boldsymbol{a},\theta)p_{\mathcal{D}}(\theta))$ for readability. Let us consider the optimization problem:

$$\operatorname{argmin}_{q_\tau} \mathcal{J} = \operatorname{argmin}_{q_\tau} \mathbb{E}_{q_\tau}\left[-\log p(\mathcal{O}|\tau) + \alpha \log q_{\boldsymbol{a}}\right]. \tag{14}$$

By applying mirror descent [27], the iterative update law of $q_\tau^{(j+1)}$ is given as

$$q_\tau^{(j+1)} = \operatorname{argmin}_{q_\tau} \langle \nabla_{q_\tau}\mathcal{J}, q_\tau\rangle + \beta \cdot \mathrm{KL}(q_\tau||q_\tau^{(j)}) + \gamma\left(1 - \int q_\tau \cdot d\tau d\theta\right), \tag{15}$$

where $\langle\cdot,\cdot\rangle$ is the inner-product operator, β is a hyper-parameter related to the step-size, and γ is the Lagrange multiplier for the constraint $\int q_\tau \cdot d\tau d\theta = 1$. The arguments in the argmin operator can be rearranged as

$$\int q_\tau \cdot \left(-\log p(\mathcal{O}|\tau) + \alpha \log q_{\boldsymbol{a}} + \beta \log q_{\boldsymbol{a}} - \beta \log q_{\boldsymbol{a}}^{(j)} - \gamma\right) d\tau d\theta + \gamma, \tag{16}$$

where, we used the relations:

$$\langle \nabla_{q_\tau}\mathcal{J}, q_\tau\rangle = \mathcal{J}, \tag{17}$$

$$\mathrm{KL}(q_\tau||q_\tau^{(j)}) = \int q_\tau \log \frac{q_\tau}{q_\tau^{(j)}} d\tau d\theta = \int q_\tau \log \frac{q_{\boldsymbol{a}}}{q_{\boldsymbol{a}}^{(j)}} d\tau d\theta. \tag{18}$$

The integrand of (16) can be organized as

$$q_\tau \cdot \log \frac{q_{\boldsymbol{a}}^{\alpha+\beta}}{p(\mathcal{O}|\tau)e^{-\gamma}(q_{\boldsymbol{a}}^{(j)})^\beta} \propto q_\tau \cdot \log \frac{q_{\boldsymbol{a}}}{p(\mathcal{O}|\tau)^{\frac{1}{\alpha+\beta}} \cdot e^{\frac{-\gamma}{\alpha+\beta}} \cdot (q_{\boldsymbol{a}}^{(j)})^{\frac{\beta}{\alpha+\beta}}} \tag{19}$$

$$= q_\tau \cdot \log \frac{p(\boldsymbol{s}|\boldsymbol{a},\theta)p_{\mathcal{D}}(\theta)q_{\boldsymbol{a}}}{(p(\boldsymbol{s}|\boldsymbol{a},\theta)p_{\mathcal{D}}(\theta)q_{\boldsymbol{a}}^{(j)}) \cdot p(\mathcal{O}|\tau)^{\frac{1}{\alpha+\beta}} \cdot e^{\frac{-\gamma}{\alpha+\beta}} \cdot (q_{\boldsymbol{a}}^{(j)})^{\frac{-\alpha}{\alpha+\beta}}} \tag{20}$$

$$= q_\tau \cdot \log \frac{q_\tau}{q_\tau^{(j)} \cdot p(\mathcal{O}|\tau)^{\frac{1}{\alpha+\beta}} \cdot e^{\frac{-\gamma}{\alpha+\beta}} \cdot (q_{\boldsymbol{a}}^{(j)})^{\frac{-\alpha}{\alpha+\beta}}}. \tag{21}$$

Integrating the above equation yields,

$$(16) = \mathrm{KL}(q_\tau||q_\tau^{(j)} \cdot p(\mathcal{O}|\tau)^{\frac{1}{\alpha+\beta}} \cdot e^{\frac{-\gamma}{\alpha+\beta}} \cdot (q_{\boldsymbol{a}}^{(j)})^{\frac{-\alpha}{\alpha+\beta}}) + \gamma. \tag{22}$$

By minimizing this equation, we get:

$$q_\tau^{(j+1)} = q_\tau^{(j)} \cdot p(\mathcal{O}|\tau)^{\frac{1}{\alpha+\beta}} \cdot e^{\frac{-\gamma}{\alpha+\beta}} \cdot (q_{\boldsymbol{a}}^{(j)})^{\frac{-\alpha}{\alpha+\beta}}. \tag{23}$$

The Lagrange multiplier can be removed using the constraint $\int q_\tau^{(j+1)} \cdot d\tau d\theta = 1$:

$$e^{\frac{\gamma}{\alpha+\beta}} = \mathbb{E}_{q_\tau^{(j)}}\left[p(\mathcal{O}|\tau)^{\frac{1}{\alpha+\beta}} \cdot (q_{\boldsymbol{a}}^{(j)})^{\frac{-\alpha}{\alpha+\beta}}\right] \tag{24}$$

$$= \mathbb{E}_{\boldsymbol{a}\sim q_{\boldsymbol{a}}{}^{(j)}}\left[\underbrace{\mathbb{E}_{\boldsymbol{s}\sim p(\boldsymbol{s}|\boldsymbol{a},\theta),\theta\sim p_{\mathcal{D}}(\theta)}\left[p(\mathcal{O}|\tau)^{\frac{1}{\alpha+\beta}}\right]}_{(*)} \cdot (q_{\boldsymbol{a}}^{(j)})^{\frac{-\alpha}{\alpha+\beta}}\right]. \tag{25}$$

Considering the discussion in Sec. 2.2 and Sec. B, we compute $(*)$ as

$$(*) \simeq f(\mathbb{E}[r(\tau)])^{\frac{1}{\alpha+\beta}} = \mathcal{W}'(\boldsymbol{a})^{\frac{1}{\alpha+\beta}}. \tag{26}$$

Substituting (25) to (23) results in:

$$q_\tau^{(j+1)} = \frac{q_\tau^{(j)} \cdot p(\mathcal{O}|\tau)^{\frac{1}{\alpha+\beta}} \cdot (q_{\boldsymbol{a}}^{(j)})^{\frac{-\alpha}{\alpha+\beta}}}{\mathbb{E}_{\boldsymbol{a}\sim q_{\boldsymbol{a}}^{(j)}}\left[\mathcal{W}'(\boldsymbol{a})^{\frac{1}{\alpha+\beta}} \cdot (q_{\boldsymbol{a}}^{(j)})^{\frac{-\alpha}{\alpha+\beta}}\right]}. \tag{27}$$

Marginalizing $(\boldsymbol{s},\theta)$, we finally obtain:

$$q_{\boldsymbol{a}}^{(j+1)} = \frac{q_{\boldsymbol{a}}^{(j)} \cdot \mathcal{W}'(\boldsymbol{a})^{\frac{1}{\alpha+\beta}} \cdot (q_{\boldsymbol{a}}^{(j)})^{\frac{-\alpha}{\alpha+\beta}}}{\mathbb{E}_{\boldsymbol{a}\sim q_{\boldsymbol{a}}^{(j)}}\left[\mathcal{W}'(\boldsymbol{a})^{\frac{1}{\alpha+\beta}} \cdot (q_{\boldsymbol{a}}^{(j)})^{\frac{-\alpha}{\alpha+\beta}}\right]}. \tag{28}$$

In (5), we replaced $\lambda := \alpha + \beta$, $\kappa := \alpha/(\alpha+\beta)$.

D Complete Definition of PaETS

$$\eta_m(\boldsymbol{a}_k) := \pi_m^{(j)} \mathcal{N}(\boldsymbol{a}_k; \boldsymbol{\mu}_m^{(j)}, \boldsymbol{\Sigma}_m^{(j)}) \Big/ \sum_{m'=1}^{M} \pi_{m'}^{(j)} \mathcal{N}(\boldsymbol{a}_k; \boldsymbol{\mu}_{m'}^{(j)}, \boldsymbol{\Sigma}_{m'}^{(j)}) \tag{29}$$

$$\omega_{m,k}^{(j+1)} := \eta_m(\boldsymbol{a}_k) w_k^{(j+1)} \Big/ \underbrace{\sum_{k'=1}^{K} \eta_m(\boldsymbol{a}_{k'}) w_{k'}^{(j+1)}}_{:=N_m} \tag{30}$$

$$\boldsymbol{\mu}_m^{(j+1)} \leftarrow \sum_{k=1}^{K} \omega_{m,k}^{(j+1)} \boldsymbol{a}_k \tag{31}$$

$$\boldsymbol{\Sigma}_m^{(j+1)} \leftarrow \sum_{k=1}^{K} \omega_{m,k}^{(j+1)} (\boldsymbol{a}_k - \boldsymbol{\mu}_m^{(j+1)})^2 \tag{32}$$

$$\pi_m^{(j+1)} \leftarrow N_m \Big/ \sum_{m=1}^{M} N_m. \tag{33}$$

E Optimization of Toy Objective Function by PaETS

Fig. 8 illustrates how PaETS optimizes $q(\boldsymbol{a}; \phi^{(j)})$ in a toy multimodal objective function.

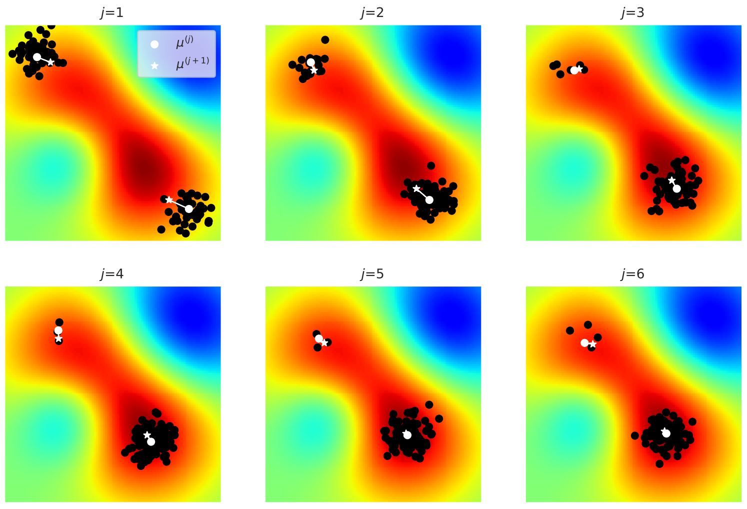


Figure 8: The optimization process of a 2D multimodal objective function by PaETS (`VIMPC('MPPI', 'GMM(M=2)', True)`), in which two distribution components are successfully optimized to fit the two modals. • depict particles that approximates $q(\boldsymbol{a}; \phi^{(j)})$.

F Computational Complexity

The main computational bottleneck of PaETS (and PETS) is the execution of ℓ3–6 in Alg. 1, in which total $K \times P$ trajectories must be sampled. In our experiment, K and P were respectively set as $K = 500$, $P = 20$ as in [13]. Compared to PETS, PaETS requires additional procedures like action sampling from GMM (ℓ2) and GMM parameter update (ℓ9). However, these additional procedures are easily parallelizable on GPUs, and their computation times are much shorter than the above mentioned bottleneck. In the experiments with our early prototype in TensorFlow, it took about 57 ms for $M = 5$ and 55 ms for $M = 1$ (equivalent to PETS) to execute one iteration of the for-loop in Alg. 1 on a single NVIDIA RTX2080 GPU. The above execution time does not meet the real-time constraints (e.g., 30 Hz). However, considering the success of the real-time implementation of MPPI in [18, 8], we believe real-time implantation of our method is feasible with optimized implementation using compiled language, low-level GPU APIs, and thorough tuning of hyperparameters (e.g., K, P and DNN complexity).

G Implementation Notes

Cross Entropy Method It is general technique to adaptively determine r_{thd} in Table 1 so that only the top-$e\%$ samples satisfies the threshold condition. We employ this technique and the *eliteness* ratio is set to be $e = 10\%$. λ has no effect on CEM optimization since $f(\cdot)$ takes binary values.

MPPI Reward normalization heuristics, as suggested in [40], were also introduced for our MPPI implementation as

$$\mathcal{W}'(\boldsymbol{a}_k)^{\frac{1}{\lambda}} = \exp\left\{\frac{1}{\lambda}\cdot\frac{r(\tau_k) - \min\{r(\tau_{k'})\}_{k'=1}^{K}}{\max\{r(\tau_{k'})\}_{k'=1}^{K} - \min\{r(\tau_{k'})\}_{k'=1}^{K}}\right\}, \tag{34}$$

where $r(\tau_k) = \frac{1}{P}\sum_{i=1}^{P} r(\tau_{k,i})$. λ was set to be $\lambda = 0.1$ as also suggested in [40].

Entropy Regularization The value of κ is very sensitive to task settings, especially for the dimensionalities of action spaces. To make κ insensitive, we introduced the following normalization trick inspired by the above heuristics. First, we rearrange (8) as

$$w_k^{(j+1)} \propto \mathcal{W}'(\boldsymbol{a})^{\frac{1}{\lambda}} \exp\left\{\kappa\cdot(-\log q^{(j)}(\boldsymbol{a}_k))\right\}. \tag{35}$$

Then, we replace $-\log q^{(j)}(\boldsymbol{a}_k)$ to normalized one:

$$-\log q^{(j)}(\boldsymbol{a}_k) \to \frac{-\log q^{(j)}(\boldsymbol{a}_k) - \min\{-\log q^{(j)}(\boldsymbol{a}_{k'})\}_{k'=1}^{K}}{\max\{-\log q^{(j)}(\boldsymbol{a}_{k'})\}_{k'=1}^{K} - \min\{-\log q^{(j)}(\boldsymbol{a}_{k'})\}_{k'=1}^{K}} \in [0,1]. \tag{36}$$

By applying these heuristics, the range of entropy bonus is limited to $[1, e^{\kappa}]$, where the action with the lowest probability among K samples gains the highest entropy bonus of e^{κ}.

H Experimental Setup

We used MuJoCo tasks modified from standard OpenAI Gym tasks.[4] Table 3 summarizes the task settings, where v_x, φ and z respectively denote the velocity, orientation angle, and height of the agents. Penalty functions Φ, Ψ are newly introduced to encourage the agents to move forward in the proper form. Instead, `done` flags used originally for early task stopping are removed. Φ, Ψ are defined as

$$\Phi(z, z_{des}) = e^{-(z-z_{des})^2}, \tag{37}$$

$$\Psi(\varphi) = \frac{1+\cos(2\varphi)}{2}. \tag{38}$$

We modified the range of actions (i.e., torques) from $[-1, 1]$ to $[-5, 5]$ to exaggerate uncertainties in the optimal trajectory posteriors.

Table 3: MuJoCo task settings.

Task	Reward Function	$\boldsymbol{s}_t \in$	$\boldsymbol{a}_t \in$	Misc.
HalfCheetah	$v_x \cdot \frac{1+\text{sign}(\cos(\varphi))}{2} - 0.1\cdot\|\|\boldsymbol{a}_t\|\|^2$	$\mathbb{R}^{18}$	$\mathbb{R}^{6}$	—
Ant	$v_x \cdot \Phi(z, z_{des}) - 10^{-3}\cdot\|\|\boldsymbol{a}_t\|\|^2$	$\mathbb{R}^{28}$	$\mathbb{R}^{8}$	$z_{des} = 0.75$
Hopper	$v_x \cdot \Phi(z, z_{des})\cdot\Psi(\varphi) - 10^{-3}\cdot\|\|\boldsymbol{a}_t\|\|^2$	$\mathbb{R}^{12}$	$\mathbb{R}^{3}$	$z_{des} = 1.2$
Walker2d	$v_x \cdot \Phi(z, z_{des})\cdot\Psi(\varphi) - 10^{-3}\cdot\|\|\boldsymbol{a}_t\|\|^2$	$\mathbb{R}^{18}$	$\mathbb{R}^{6}$	$z_{des} = 1.2$

Table 4 summarizes the shared parameter settings for MBRL (PaETS, PETS, and MPPI). For SAC, we used the default parameters from the original codebase.

[4] https://github.com/openai/gym

Table 4: MBRL parameters.

	HalfCheetah	Ant	Hopper	Walker2d
T: prediction horizon	30	30	60	45
κ: weight of entropy regularizer	0.5	0.25	0.5	0.5
K: # sampled actions	500			
P: # trajectories for each action	20			
U: # optimization-iterations	5			
H: # episode length	1000			
E: # neural networks	5			
hidden nodes	(200, 200, 200, 200)			
activation function	Swish			
optimizer	Adam			
learning rate	10^{-3}			
batch-size	160			

I Diversity Analysis of $\mathcal{D}$

In this section, we analyzes the diversity of training data $\mathcal{D}$ collected by different MPC-policies. The distributions (histograms) of the data samples are illustrated in Fig. 9, in which the dimension of a sample $(\boldsymbol{s}, \boldsymbol{a})$ was reduced by t-SNE. This figure suggests that incorporating uncertainty both in the dynamics and optimalities can improve the diversity of $\mathcal{D}$ (i.e., coverage of state-action space).

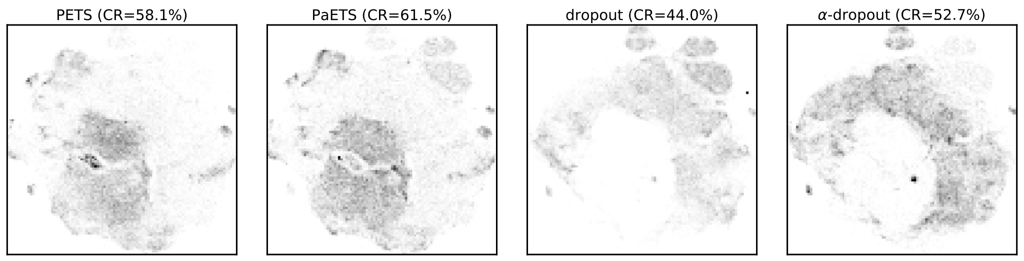


Figure 9: Comparison of training data distributions collected by different MPC-policies. CR (cover ratio) indicates the ratio of non-zero bins in each 2D histogram.