2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

PlaNet of the Bayesians: Reconsidering and Improving Deep Planning Network by Incorporating Bayesian Inference

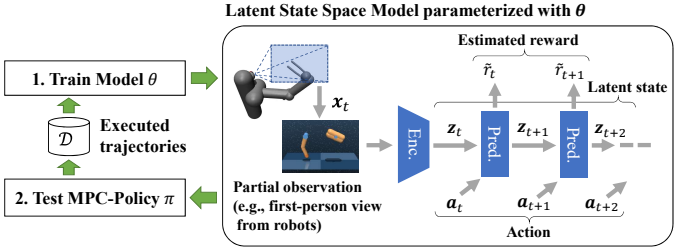
Masashi Okada[†,⋆], Norio Kosaka[†] and Tadahiro Taniguchi[†,*]

***Abstract*— In the present paper, we propose an extension of the Deep Planning Network (PlaNet), also referred to as PlaNet of the Bayesians (PlaNet-Bayes). There has been a growing demand in model predictive control (MPC) in partially observable environments in which complete information is unavailable because of, for example, lack of expensive sensors. PlaNet is a promising solution to realize such latent MPC, as it is used to train state-space models via model-based reinforcement learning (MBRL) and to conduct planning in the latent space. However, recent state-of-the-art strategies mentioned in MBRR literature, such as involving uncertainty into training and planning, have not been considered, significantly suppressing the training performance. The proposed extension is to make PlaNet uncertainty-aware on the basis of Bayesian inference, in which both model and action uncertainty are incorporated. Uncertainty in latent models is represented using a neural network ensemble to approximately infer model posteriors. The ensemble of optimal action candidates is also employed to capture multimodal uncertainty in the optimality. The concept of the action ensemble relies on a general variational inference MPC (VI-MPC) framework and its instance, probabilistic action ensemble with trajectory sampling (PaETS). In this paper, we extend VI-MPC and PaETS, which have been originally introduced in previous literature, to address partially observable cases. We experimentally compare the performances on continuous control tasks, and conclude that our method can consistently improve the asymptotic performance compared with PlaNet.**

I. Introduction

In the present paper, we focus on model predictive control (MPC) in partially observable environments. MPC is a promising technique used in advanced control systems that relies on the specified system models to predict future states and rewards for the purpose of planning. The clear explainability of such decision-making processes is preferable, especially for industrial systems, and therefore, many real-world applications have introduced MPC such as HVAC systems [1], manufacturing processes [2], and power electronics [3]. The above systems assume fully observable environments; however, in practical applications, complete information is often unavailable, as measuring sufficient information for planning may be difficult and/or require expensive devices (for example, LiDARs).

The Deep Planning Network (PlaNet) [4] is an MPC-oriented model-based reinforcement learning (MBRL) method used for partially observable environments. Figure 1 outlines the concept of PlaNet in which an encoder is utilized to convert partial observations $\boldsymbol{x}_t$ (e.g., high-dimensional raw observations such as images) into latent states $\boldsymbol{z}_t$. In the latent space, a prediction model is used to estimates next latent states $\boldsymbol{z}_{t+1}$ and rewards $\tilde{r}_t$ based on current states $\boldsymbol{z}_t$ and given actions $\boldsymbol{a}_t$. This modeling way allows planning in the latent state-space in which we expect that the sufficient information for planning is embedded. As our primary interest is to apply MPC to practical systems, in this paper, we mainly focus on PlaNet and consider this method as a strong baseline.



Utilize Bayesian inference?

	1. Latent Model θ	2. MPC-Policy π
PlaNet [Hafner et al., 2019]	✗: $\mathrm{argmax}_\theta\, p(\mathcal{D}\|\theta)$	✗: $\mathrm{argmax}_{\boldsymbol{a}_{\geq t}}\, r(\boldsymbol{z}_{\geq t}, \boldsymbol{a}_{\geq t})$
PlaNet-Bayes (Ours)	✓: Infer $p(\theta\|\mathcal{D})$	✓: Infer $p(\boldsymbol{z}_{\geq t}, \boldsymbol{a}_{\geq t}\|\mathcal{O}_{\geq t})$

Fig. 1. Top: the concept of latent state-space model training and planning in the learned latent space. Buttom: comparison of PlaNet [4] and the proposed PlaNet-Bayes. $\mathcal{O}$ indicates a random variable called *optimality*, which is introduced later in Sec. II-B.

Recent studies on MBRL in fully observable environments have shown that uncertainty-aware modeling is essential to enhance the training performance. The *model-bias problem* [5], which occurs because of the overfitting of model parameters θ with respect to the limited amount of data $\mathcal{D}$ available during an early training phase, has been regarded as an inherent problem that restrains the MBRL potential. However, recent studies have demonstrated that incorporating uncertainty in the model parameters can alleviate this issue by exploiting Bayesian inference of posterior $p(\theta|\mathcal{D})$ [5]–[13]. In addition, in [12], they have shown that involving uncertainty in optimal actions also affects performance improvement. In the literature, the trajectory optimization problem in the Markov decision process is formulated as a variational inference problem [14], deriving a general framework called variational inference MPC (VI-MPC). As an instance of the framework, probabilistic action ensemble with trajectory sampling (PaETS) is also proposed that uses Gaussian mixture model (GMM) as the variational distri-

† Masashi Okada, Norio Kosaka, and Tadahiro Taniguchi are with AI Solutions Center, Business Innovation Division, Panasonic Corporation, Japan.
* Tadahiro Taniguchi is also with Ritsumeikan University, College of Information Science and Engineering, Japan.
⋆ Corresponding author: okada.masashi001@jp.panasonic.com

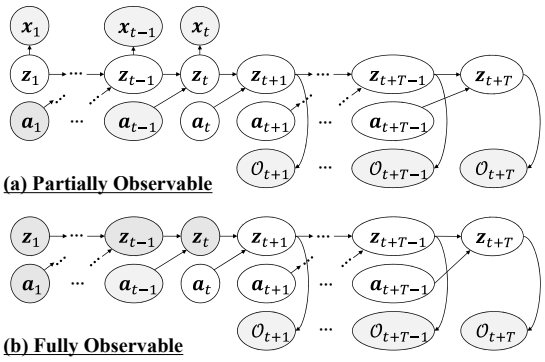


Fig. 2. Graphical models discussed in this paper where t is the current time-step and T is the planning horizon. The training objective of PlaNet and VI-MPC for partially observable environments are derived from (a). VI-MPC for fully observable environments is derived from (b).

bution by using which we can naturally model multimodal uncertainty in optimal actions.

Considering the observations above, it is obvious that PlaNet is insufficiently uncertainty-aware, which leads to limitations on its strong potential. Although the training procedure employed in PlaNet is based on autoencoding variational inference [15], the model parameters are estimated as fixed-points θ, underestimating the model uncertainty $p(\theta|\mathcal{D})$. For the purpose of planning, the cross entropy method (CEM) [16] is heuristically introduced for trajectory optimization, which ignores the multimodal uncertainty in optimal actions.

Motivated by this, in the present study, we reconsider PlaNet from a Bayesian viewpoint, aiming to propose an extension of PlaNet, referred to as *PlaNet of the Bayesians* (PlaNet-Bayes). The primary contributions, which characterize PlaNet-Bayes, can be summarized as follows;

- We propose incorporating uncertainty in latent state-space models by considering approximate inference of the posterior $p(\theta|\mathcal{D})$ with a neural network ensemble.
- We formulate VI-MPC for partially observable environments by considering latent planning as variational inference. Exploiting the newly derived framework, we also introduce a latent version of PaETS to involve multimodality in optimal actions.

The differences between the proposed method and the baseline PlaNet are summarized in the bottom part of Fig. 1. By involving the two types of the uncertainty, PlaNet-Bayes can ahieve the performance consistently better compared with PlaNet.

The remainder of this paper is organized as follows. In Sec. II, we provide a brief review on PlaNet, VI-MPC, and PaETS. In Sec. III, we describe the proposed PlaNet-Bayes in detail. In Sec. IV, the effectiveness of PlaNet-Bayes is demonstrated through evaluations using the DeepMind control suite [17].

II. Preliminary

A. PlaNet: Deep Planning Network

1) Autoencoding Variational Bayes for Time Series: Let us begin with considering the graphical model illustrated in Fig. 2(a). In this section, we focus on $\boldsymbol{z}_{\leq t}$ and their adjacent nodes. The remaining nodes, namely $\boldsymbol{z}_{>t}$, $\boldsymbol{a}_{\geq t}$, and $\mathcal{O}_{>t}$, are discussed later in Sec. III-B.1. The joint distribution of the focused variables is defined as follows:

$$p_{\text{joint}}(\boldsymbol{z}_{\leq t}, \boldsymbol{a}_{<t}, \boldsymbol{x}_{\leq t}) = \underbrace{p(\boldsymbol{z}_1)\left\{\prod_{t'=1}^{t-1} p\left(\boldsymbol{z}_{t'+1}|\boldsymbol{z}_{t'}, \boldsymbol{a}_{t'}\right)\right\}}_{\coloneqq p(\boldsymbol{z}_{\leq t}|\boldsymbol{a}_{<t})} \cdot \underbrace{\left\{\prod_{t'=1}^{t} p\left(\boldsymbol{x}_{t'}|\boldsymbol{z}_{t'}\right)\right\}}_{\coloneqq p(\boldsymbol{x}_{\leq t}|\boldsymbol{z}_{\leq t})}. \quad (1)$$

As in the case of well-known variational autoencoders (VAEs) [15], generative models $p(\boldsymbol{z}_{t+1}|\boldsymbol{z}_t, \boldsymbol{a}_t)$, $p(\boldsymbol{x}_t|\boldsymbol{z}_t)$ and inference model $q(\boldsymbol{z}_t|\boldsymbol{x}_{\leq t}, \boldsymbol{a}_{<t})$ can be trained by maximizing the evidence lower bound (ELBO):

$$\begin{aligned}
&\log p(\boldsymbol{x}_{\leq t}|\boldsymbol{a}_{\leq t}) = \log \int p(\boldsymbol{z}_{\leq t}|\boldsymbol{a}_{<t}) p(\boldsymbol{x}_{\leq t}|\boldsymbol{z}_{\leq t}) d\boldsymbol{z}_{\leq t} \\
&\geq \underbrace{\mathbb{E}_{q(\boldsymbol{z}_{\leq t}|\boldsymbol{x}_{\leq t}, \boldsymbol{a}_{\leq t})}\left[\log p(\boldsymbol{x}_{\leq t}|\boldsymbol{z}_{\leq t})\right]}_{\text{reconstruction}} \hookleftarrow \\
&\quad - \underbrace{D_{\text{KL}}\left[q(\boldsymbol{z}_{\leq t}|\boldsymbol{x}_{\leq t}, \boldsymbol{a}_{\leq t}) || p(\boldsymbol{z}_{\leq t}|\boldsymbol{a}_{<t})\right]}_{\text{complexity}}, \quad (2)
\end{aligned}$$

where,

$$q(\boldsymbol{z}_{\leq t}|\boldsymbol{x}_{\leq t}, \boldsymbol{a}_{\leq t}) \coloneqq \prod_{t'=1}^{t} q(\boldsymbol{z}_{t'}|\boldsymbol{x}_{\leq t'}, \boldsymbol{a}_{<t'}). \quad (3)$$

If the models are parameterized with θ, this objective can be maximized by the stochastic gradient ascent via backpropagation. For the purpose of latent planning, a reward function $p(r_t|\boldsymbol{z}_t)$ is also required to be formulated. To do this, we can simply regard the rewards as observations and learn the reward function along with $p(\boldsymbol{x}_{\boldsymbol{z}}|\boldsymbol{z}_t)$.

2) Recurrent State-Space Model: PlaNet introduces Recurrent State-Space Model (RSSM), which assumes the latent $\boldsymbol{z}_t$ which comprises $\boldsymbol{z}_t = (\boldsymbol{s}_t, \boldsymbol{h}_t)$ where $\boldsymbol{s}_t$, $\boldsymbol{h}_t$ are the probabilistic and deterministic variable, respectively. The generative and inference models can be formulated as:

$$\begin{aligned}
&\text{Generative models}: \begin{cases} \boldsymbol{h}_t = f^{\text{GRU}}(\boldsymbol{h}_{t-1}, \boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1}) \\ \boldsymbol{s}_t \sim p(\boldsymbol{s}_t|\boldsymbol{h}_t) \\ \boldsymbol{x}_t, r_t \sim p(\boldsymbol{x}_t, r_t|\boldsymbol{h}_t, \boldsymbol{s}_t) \end{cases}, \\
&\text{Inference model}: \boldsymbol{s}_t \sim q(\boldsymbol{s}_t|\boldsymbol{h}_t, \boldsymbol{x}_t), \quad (4)
\end{aligned}$$

where deterministic $\boldsymbol{h}_t$ is considered as the internal state of the gated recurrent unit (GRU) $f^{\text{GRU}}(\cdot)$ [18], so that historical information is embedded into $\boldsymbol{h}_t$. The architectures of the generative and inference models are illustrated in Fig. 3.

3) Latent Planning using RSSM: Algorithm 1 outlines PlaNet's latent planning strategy with CEM [16]. CEM is a stochastic method based on importance sampling that is used to iteratively update a proposal distribution to optimize an objective function by executing the following steps: (1) sample K candidates from the proposal distribution, (2) then evaluate the objective function for each sample, and

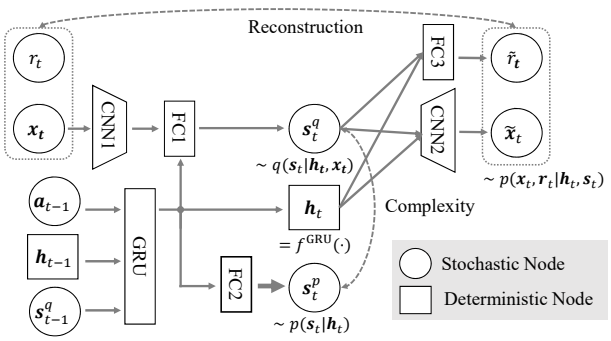


Fig. 3. The architecture of RSSM. CNN, GRU and FC represent a convolutional neural network, a GRU-cell, and a fully-connected layer, respectively.

(3) update the proposal distribution using the results of evaluation. Algorithm 1 summarizes the process executed by PlaNet to implements the above steps for latent planning. (1) At $\ell 5$, K candidate action sequences are sampled from the diagonal Gaussian proposal parameterized with mean $\boldsymbol{\mu}_{t:t+T-1}$ and variance $\boldsymbol{\sigma}_{t:t+T-1}$. In the pseudo-code, the subscript $\square_{t:t+T-1}$ is omitted for readability. (2) Starting from the current state $(\boldsymbol{h}_t, \boldsymbol{s}_t)$ estimated at ℓ1–3, latent trajectories and step rewards are sampled up to T time-steps at ℓ6–10. (3) Using the trajectory rewards calculated at $\ell 11$, the parameters $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$ are adjusted according to the CEM update law at ℓ12–14, where $\mathbb{1}[\cdot]$ is an indicator function and R_{thd} is determined so that top-$e\%$ of samples satisfy the threshold condition ($e = 10\%$ is used in [4]). This iterative update is executed U times.

Algorithm 1: PlaNet's latent planning based on CEM [4]

Input: Current observation $\boldsymbol{x}_t$,
Previous latent states and action $(\boldsymbol{h}_{t-1}, \boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1})$,
Initial parameters of the proposal distribution $(\boldsymbol{\mu}^{(0)}, \boldsymbol{\sigma}^{(0)})$
Output: Optimized parameters of proposal dist. $(\boldsymbol{\mu}^{(U)}, \boldsymbol{\sigma}^{(U)})$
Current latent states $(\boldsymbol{h}_t, \boldsymbol{s}_t)$

// (0) Estimate current latent state
if $t > 1$ **then**
  Evolve the latent state $\boldsymbol{h}_t = f^{\mathrm{GRU}}(\boldsymbol{h}_{t-1}, \boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1})$
Infer the latent state $\boldsymbol{s}_t \sim q(\boldsymbol{s}_t|\boldsymbol{h}_t, \boldsymbol{x}_t)$
for $j \leftarrow 1$ **to** U **do**
  // (1) Sample K candidates
  Sample actions $\{\boldsymbol{a}_k \sim q(\boldsymbol{a}; \boldsymbol{\mu}^{(j-1)}, \boldsymbol{\sigma}^{(j-1)}\}_{k=1}^{K}$
  // (2) Eval. the objective for each candidate
  Sample trajectories and rewards $\{\{$
    $\boldsymbol{h}_{k,t'+1} = f^{\mathrm{GRU}}(\boldsymbol{h}_{t'}, \boldsymbol{s}_{t'}, \boldsymbol{a}_{k,t'})$,
    $\boldsymbol{s}_{k,t'+1} \sim p(\boldsymbol{s}_{t'+1}|\boldsymbol{h}_{k,t'+1})$
    $r_{k,t'+1} \sim p(r_{t'+1}|\boldsymbol{h}_{k,t'+1}, \boldsymbol{s}_{k,t'+1})$
  $\}_{t'=t}^{t+T-1}\}_{k=1}^{K}$
  Calc. trajectory rewards $\{R_k = \sum_{t'=t+1}^{t+T} r_{k,t'}\}_{k=1}^{K}$
  // (3) Update proposal distribution
  Calc. weights $\{w_k = \mathbb{1}[R_k \geq R_{\mathrm{thd}}]\}_{k=1}^{K}$
  Normalize weights $\{w_k \leftarrow w_k / \sum_{k'=1}^{K} w_{k'}\}_{k=1}^{K}$
  Update $(\boldsymbol{\mu}^{(j)}, (\boldsymbol{\sigma}^{(j)})^2) \leftarrow$
    $(\sum_{k=1}^{K} w_k \cdot \boldsymbol{a}_k, \sum_{k=1}^{K} w_k \cdot (\boldsymbol{a}_k - \boldsymbol{\mu}^{(j)})^2)$

B. VI-MPC and PaETS in Fully Observable Environments

Figure 2(b) represents the graphical model that is used to derive original VI-MPC [12] based on the *control as inference* framework [14]. To formulate optimal control as inference, a binary random variable $\mathcal{O}_{t'} \in \{0, 1\}$ is auxiliarily introduced to represent the *optimality* of state $\boldsymbol{z}_{t'}$. Note that $\boldsymbol{z}_{t'}$ is not *latent* state here. Let us consider the trajectory posterior conditioned on the optimality $p(\boldsymbol{a}_{\geq t}, \boldsymbol{z}_{>t}|\mathcal{O}_{>t})$. Hereinafter, we denote $\boldsymbol{a}_{\geq t}$, $\boldsymbol{z}_{>t}$ and $\mathcal{O}_{>t}$ as; $\boldsymbol{a}$, $\boldsymbol{z}$ and $\mathcal{O}$ for readability. By solving a variational inference problem: $\mathrm{argmin}_q \, \mathrm{KL}(q(\boldsymbol{a}, \boldsymbol{z})||p(\boldsymbol{a}, \boldsymbol{z}|\mathcal{O}))$, we obtain the iterative law to update the variational distribution q as per equation bellow:

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

where j indicates the loop count, λ is the inverted step-size to control optimization speed, and κ is the weight of the entropy regularization term $q^{-\kappa}$. $\mathcal{W}$ is defined as:

$$\mathcal{W}(\boldsymbol{a}) \coloneqq \mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{a})}[p(\mathcal{O}|\boldsymbol{z})]. \tag{6}$$

This is a general framework called VI-MPC that generalizes several MPC methods. Different definitions of optimality likelihood $p(\mathcal{O}|\boldsymbol{z})$ recover various methods including CEM, path integral control [19], [20], covariance matrix adaptation evolution strategy (CMA-ES) [21], and proportional CEM [22]. The above-mentioned methods generally assume that q is Guassian. However, VI-MPC can define q arbitrarily, and PaETS defines q as GMM successfully incorporating multimodality of optimal actions. The original formulation of VI-MPC supposes fully observable environments, and application to partially observable cases has not been discussed.

III. PlaNet-Bayes: PlaNet of the Bayesians

This section describes an extension of PlaNet referred to as *PlaNet of the Bayesians* (PlaNet-Bayes) that implies incorporating two types of uncertainty in latent models and actions by exploiting Bayesian inference. The process of incorporating both types of uncertainty is introduced in Sections III-A and III-B.

A. Incorporating Model Uncertainty

In general, VAEs denote generative and inference models like $p_\theta(\cdot)$, $q_\theta(\cdot)$, assuming a point estimation of the model parameters θ. Here, we remove this assumption and treat θ as a random variable. As indicated in Eq. 2, maximizing ELBO also leads to maximization of $p(\mathcal{D}|\theta)$, and therefore we can regard VAEs' general training procedure as a maximum likelihood estimation. Instead, by inferring the posterior $p(\theta|\mathcal{D})$, we incorporate the uncertainty in the model parameters of RSSM. Given a sufficiently parameterized model, i.e., a deep neural network, promising schemes for approximating the posterior are stochastic gradient MCMC [23], *dropout as variational inference* [6]–[8], and neural network ensembles [9]–[12].

In this paper, we use the ensemble scheme owing to its simplicity and better performance compared with dropout [12]. This scheme is employed to approximate the posterior as a set of *particles* $p(\theta|\mathcal{D}) \simeq \frac{1}{E}\sum_i^E \delta(\theta - \theta_i)$, where E is the ensemble size (namely, the number of

networks) and δ is Dirac delta function. This approximation can successfully incorporate multimodal uncertainty in the exact posterior. Each particle θ_i is independently trained by stochastic gradient descent so as to (sub-)optimize $p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta)$. In the proposed modeling scheme, `GRU`, `FC1`, and `FC2`, as presented in Fig. 3, are implemented as ensemble networks, and the ensemble size is set to be $E = 5$ as same as defined in [9], [12].

B. Incorporating Action Uncertainty

1) Derivation of latent VI-MPC: We derive VI-MPC for partially observable environments by formulating latent planning as a variational inference problem. We refer to the graphical model of Fig. 2(a) again for this formulation. For the purpose of clarity, in this figure, we omit the random variable θ using which the latent state transition is conditioned, namely, $p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{a}_{t-1}, \theta)$. The joint distribution of all variables in this figure is:

$$p_{\text{joint}}(\boldsymbol{z}, \boldsymbol{a}, \theta, \mathcal{O}_{\geq t}, \boldsymbol{x}_{\leq t}) = p(\mathcal{O}_{>t}|\boldsymbol{z}_{>t}) \cdot p(\boldsymbol{z}|\boldsymbol{a}, \theta) \cdot p(\boldsymbol{x}_{\leq t}|\boldsymbol{z}_{\leq t}) \cdot p(\theta|\mathcal{D}), \quad (7)$$

where,

$$p(\mathcal{O}_{>t}|\boldsymbol{z}_{>t}) = \prod_{t'=t+1}^{t+T} p(\mathcal{O}_{t'}|\boldsymbol{z}_{t'}), \quad (8)$$

and a non-informative action prior $p(\boldsymbol{a})$ is supposed. The subscripts of $\boldsymbol{z}_{1:t+T}$, $\boldsymbol{a}_{1:t+T-1}$ are omitted. The objective of this formulation is to infer the posterior:

$$p(\boldsymbol{a}_{\geq t}, \boldsymbol{z}, \theta|\mathcal{O}_{\geq t}, \boldsymbol{x}_{\leq t}) = \frac{p_{\text{joint}}(\cdot)}{\int p_{\text{joint}}(\cdot) d\boldsymbol{a}_{\geq t} d\boldsymbol{z} d\theta}. \quad (9)$$

As the inference of this posterior is intractable, instead, we estimate the variational distribution $q(\boldsymbol{a}_{\geq t}, \boldsymbol{z}, \theta)$ that minimizes $\mathrm{KL}(q(\boldsymbol{a}_{\geq t}, \boldsymbol{z}, \theta)||p(\boldsymbol{a}_{\geq t}, \boldsymbol{z}, \theta|\mathcal{O}_{>t}, \boldsymbol{x}_{\leq t}))$. We suppose that q is factorized as:

$$q(\boldsymbol{a}_{\geq t}, \boldsymbol{z}, \theta) = q(\boldsymbol{a}_{\geq t}) p(\boldsymbol{z}|\boldsymbol{a}, \theta) p(\boldsymbol{x}_{\leq t}|\boldsymbol{z}_{\leq t}) p(\theta|\mathcal{D}). \quad (10)$$

This variational inference problem can be solved by maximizing the ELBO:

$$\log p(\mathcal{O}_{>t}, \boldsymbol{x}_{\leq t}) \geq \mathbb{E}_{q(\boldsymbol{a}_{\geq t}), p(\boldsymbol{z}|\boldsymbol{a}, \theta), p(\theta|\mathcal{D})}[\hookleftarrow p(\boldsymbol{x}_{\leq t}|\boldsymbol{z}_{\leq t}) p(\mathcal{O}_{\geq t}|\boldsymbol{z}_{\geq t}) - q(\boldsymbol{a}_{\geq t})]. \quad (11)$$

By applying the mirror descent [12], [24], [25] to this optimization problem, we can derive an update law for $q(\boldsymbol{a}_{\geq t})$, which takes the same form as Eq. 5. At the same time, $\mathcal{W}(\boldsymbol{a}_{\geq t})$ is defined differently from the original formulation:

$$\mathcal{W}(\boldsymbol{a}_{\geq t}) \coloneqq \mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{a}, \theta), p(\theta|\mathcal{D})}[p(\boldsymbol{x}_{\leq t}|\boldsymbol{z}_{\leq t}) p(\mathcal{O}_{>t}|\boldsymbol{z}_{>t})]. \quad (12)$$

A major difference from the original VI-MPC is that the observation likelihood $p(\boldsymbol{x}_{\leq t}|\boldsymbol{z}_{\leq t})$ should be considered in Eq. 12. One may consider that Eq. 12 can be efficiently implemented with Sequential Monte Carlo (SMC also referred to as particle filter) [26] using the generative models; namely, for $t' \leq t$, predict particles by $p(\boldsymbol{z}_{t'}|\boldsymbol{z}_{t'-1}, \boldsymbol{a}_{t'-1}, \theta)$ and conduct resampling with the likelihood $p(\boldsymbol{x}_{t'}|\boldsymbol{z}_{t'})$. Because of the fact that this yields rather a complicated process, let us consider to approximate it by utilizing the inference model. By applying importance sampling with $q(\boldsymbol{z}_{\leq t}|\boldsymbol{x}_{\leq t}, \boldsymbol{a}_{\leq t}, \theta)$, Eq. 12 can be rearranged as:

$$\mathcal{W}(\boldsymbol{a}_{\geq t}) = \mathbb{E}_{\mathbb{P}}\left[\frac{p(\boldsymbol{z}_{\leq t}|\boldsymbol{a}_{<t}, \theta)}{q(\boldsymbol{z}_{\leq t}|\boldsymbol{x}_{\leq t}, \boldsymbol{a}_{\leq t}, \theta)} p(\boldsymbol{x}_{\leq t}|\boldsymbol{z}_{\leq t}) p(\mathcal{O}_{\geq t}|\boldsymbol{z}_{\geq t})\right], \quad (13)$$

where,

$$\mathbb{P} \coloneqq q(\boldsymbol{z}_{\leq t}|\boldsymbol{x}_{\leq t}, \boldsymbol{a}_{\leq t}, \theta) p(\boldsymbol{z}_{>t}|\boldsymbol{a}_{\geq t}, \theta) p(\theta|\mathcal{D}). \quad (14)$$

Having this distribution $\mathbb{P}$, $z_{\leq t}$ and $z_{>t}$ are sampled from the inference model $q(\boldsymbol{z}_t|\cdot)$ and generative model $p(\boldsymbol{z}_t|\cdot)$, respectively. As these models are trained so as to minimize the complexity loss in Eq. 2, we can expect that the likelihood ratio in Eq. 13, i.e., $p(\boldsymbol{z}_{\leq t}|\cdot)/q(\boldsymbol{z}_{\leq t}|\cdot)$, can be canceled. In addition, minimization of the reconstruction loss in Eq. 2 allow realizing successful autoencoding $\boldsymbol{x} \to \boldsymbol{z} \to \boldsymbol{x}$ so that

$$p(\boldsymbol{x}_{\leq t}|\boldsymbol{z}_{\leq t}) \simeq \text{const. if } \boldsymbol{z}_{\leq t} \sim q(\boldsymbol{z}_{\leq t}|\boldsymbol{x}_{\leq t}, \boldsymbol{a}_{\leq t}, \theta) \quad (15)$$

Therefore, the likelihood $p(\boldsymbol{x}_{\leq t}|\boldsymbol{z}_{\leq t})$ can be approximately pulled out from $\mathbb{E}_{\mathbb{P}}[\cdot]$ and considered as canceled in Eq. 5. On the basis of these observations, we rewrite Eq. 13 as:

$$\mathcal{W}(\boldsymbol{a}_{\geq t}) \coloneqq \mathbb{E}_{\mathbb{P}}\left[p(\mathcal{O}_{\geq t}|\tau_{\geq t})\right]. \quad (16)$$

Eqs. 5, 14, and 16 suggest general VI-MPC for the latent state-space models. The planning method used in PlaNet is a special case of VI-MPC, assuming that (1) the variational distribution q is Gaussian; (2) the optimality likelihood is defined as: $p(\mathcal{O}_{\geq t}|\boldsymbol{z}_{\geq t}) \coloneqq \mathbb{1}[R(\boldsymbol{z}_{\geq t}) \geq R_{\text{thd}}]$ ($R(\boldsymbol{z}_{\geq t})$: trajectory reward); (3) $\mathbb{E}_{\mathbb{P}}[\cdot]$ is approximated by a single sample; (4) $p(\theta|\mathcal{D})$ is inferred as a single point; and (5) the entropy regularizer κ is set to be $\kappa \to 0$. The generality of original VI-MPC holds in this partially observable case, and we can utilize different definitions of q and $p(\mathcal{O}_{\geq t}|\boldsymbol{z}_{\geq t})$. In this study, same optimality likelihood with PlaNet is introduced.

2) PaETS for latent planning: We introduce a latent version of PaETS using the proposed latent VI-MPC. PaETS uses GMM as the variational distribution q as:

$$q^{(j)}(\boldsymbol{a}_{\geq t}) \coloneqq q(\boldsymbol{a}_{\geq t}; \phi^{(j)}) = \sum_{m=1}^{M} \pi_m^{(j)} \mathcal{N}(\boldsymbol{a}_{\geq t}; \boldsymbol{\mu}_m^{(j)}, \boldsymbol{\sigma}_m^{(j)}), \quad (17)$$

where $\phi^{(j)} \coloneqq \{(\pi_m^{(j)}, \boldsymbol{\mu}_m^{(j)}, \boldsymbol{\sigma}_m^{(j)})\}_{m=1}^{M}$ and M is the number of components in the mixture model. By following the derivation procedure similarly as in [12], we obtain the update laws of $\phi^{(j+1)}$, which take the weight-average form like $\ell 14$ of Algorithm 1:

$$\left(\boldsymbol{\mu}_m^{(j+1)}, \boldsymbol{\sigma}_m^{(j+1)}, \pi_m^{(j+1)}\right) \leftarrow \left(\sum_k \omega_{m,k}^{(j+1)} \boldsymbol{a}_k, \sum_k \omega_{m,k}^{(j+1)} (\boldsymbol{a}_k - \boldsymbol{\mu}_m^{(j+1)})^2, \frac{N_m}{\sum_{m'=1}^{M} N_{m'}}\right). \quad (18)$$

The complete definition of Eq. 18 is available in Appendix I.

C. Procedure and Implementation

Algorithms 2 and 3 summarize the steps of the proposed method. The outermost loop of Algorithm 2 describes the training of the posterior $p(\theta|\mathcal{D})$ (namely, neural network ensemble), at which the model is iteratively trained at $\ell 3$. The trained model is tested on the MPC loop at $\ell 6$–11, whereas $\mathcal{D}$ is augmented according to the executed trajectories. This paper realizes GMM parameter initialization at $\ell 7$ as follows; $\boldsymbol{\mu}_m^{(0)}$ is reset by a general warm-start technique of MPC, and $(\boldsymbol{\sigma}_m^{(0)}, \pi_m^{(0)})$ is set to be $(\mathbf{1}/2, 1/M)$.

The procedure in Algorithm 3 is rather similar to that one of in Algorithm 1. One of the primary differences is that the expectation $\mathbb{E}_{\mathbb{P}}[\cdot]$ in Eq. 16 is approximated with E samples obtained from the E ensemble networks, whereas PlaNet uses only a single sample. Another difference is that PaETS is employed to update the variational distribution parameter ϕ at $\ell 12$[1].

Algorithm 2: PlaNet-Bayes' training and control loop

1 Initialize $\mathcal{D}$ with a random controller for several trials
// Training loop
2 **repeat**
3 Train $\{\theta_i\}_{i=1}^E$ by optimizing Eq. 2
4 Initialize $\{(\boldsymbol{h}_{i,0}, \boldsymbol{s}_{i,0})\}_{i=1}^E$ and $\boldsymbol{a}_0$ as $\mathbf{0}$s
5 Reset environment and observe $\boldsymbol{x}_1$
// Control loop, H: episode length
6 **for** $t \leftarrow 1$ **to** H **do**
7 Initialize GMM parameter $\phi^{(0)}$
8 Execute Alg. 3
9 Sample $\boldsymbol{a}_{t:t+T-1} \sim q(\boldsymbol{a}_{t:t+T-1}; \phi^{(U)})$
10 Send $\boldsymbol{a}_t$ to actuators and observe $(\boldsymbol{x}_{t+1}, r_{t+1})$
11 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{x}_t, \boldsymbol{a}_t, r_{t+1})\}$
12 **until** *the MPC-policy performs well*

We implement PlaNet-Bayes in TensorFlow [27] by modifying the official source code of PlaNet[2]. We keep the most of hyperparameters and experimental conditions similar as the original ones except for the number of action candidates K (originally $K = 1,000$). As $\mathbb{E}_{\mathbb{P}}[\cdot]$ is approximated with $E = 5$ samples, we have to sample $K \times E$ trajectories at $\ell 6$–10 in Algorithm 3, which requires E-times larger computations. To relieve this, we set $K \leftarrow K/E (= 1,000/5 = 200)$ so that the same number of trajectories are evaluated; namely the total number of multiply-accumulate operations are kept same. The other hyperparameters introduced in this papers are $\kappa = 0$, $\lambda = 1$, and $M = 5$.

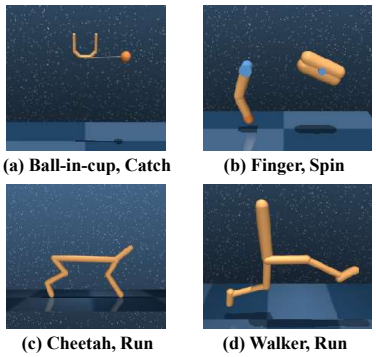
IV. Experiments

A. Comparison with PlaNet

The main objective of this experiment is to demonstrate that PlaNet-Bayes has advantages over the baseline method

Algorithm 3: PlaNet-Bayes' latent planning with PaETS

Input: Current observation $\boldsymbol{x}_t$,
Previous latent states and action $\{(\boldsymbol{h}_{i,t-1}, \boldsymbol{s}_{i,t-1})\}_{i=1}^E$, $\boldsymbol{a}_{t-1}$,
Initial parameters of variational distribution $\phi^{(0)}$
Output: Optimized parameters of variational distribution $\phi^{(U)}$
Current latent states $\{(\boldsymbol{h}_{i,t}, \boldsymbol{s}_{i,t})\}_{i=1}^E$
// (0) Estimate the current latent state
1 **if** $t > 1$ **then**
2 Evolve the latent state
$\{\boldsymbol{h}_{i,t} = f^{\mathrm{GRU}}(\boldsymbol{h}_{i,t-1}, \boldsymbol{s}_{i,t-1}, \boldsymbol{a}_{t-1}; \theta_i)\}_{i=1}^E$
3 Infer the latent state $\{\boldsymbol{s}_{i,t} \sim q(\boldsymbol{s}_t|\boldsymbol{h}_{i,t}, \boldsymbol{x}_t, \theta_i)\}_{i=1}^E$
4 **for** $j \leftarrow 1$ **to** U **do**
// (1) Sample K candidates
5 Sample actions $\{\boldsymbol{a}_k \sim q(\boldsymbol{a}; \phi^{(j-1)}\}_{k=1}^K$
// (2) Eval. the objective for each candidate
6 Sample trajectories and rewards $\{\{\{$
7 $\boldsymbol{h}_{k,i,t'+1} = f_{\theta_i}^{\mathrm{GRU}}(\boldsymbol{h}_{k,i,t'}, \boldsymbol{s}_{k,i,t'}, \boldsymbol{a}_{k,t'})$,
8 $\boldsymbol{s}_{k,i,t'+1} \sim p(\boldsymbol{s}_{k,t'+1}|\boldsymbol{h}_{k,i,t'+1}, \theta_i)$
9 $r_{k,i,t'+1} \sim p(r_{t'+1}|\boldsymbol{h}_{k,i,t'+1}, \boldsymbol{s}_{k,i,t'+1})$
10 $\}_{t'=t}^{t+T-1}\}_{i=1}^E\}_{k=1}^K$
11 Calc. trajectory rewards
$\{R_k = \frac{1}{E}\sum_{i=1}^E \sum_{t'=t+1}^{t+T} r_{k,i,t'}\}_{k=1}^K$
// (3) Update proposal distribution
12 Update $\phi^{(j)}$ by Eq. 18.



(a) Ball-in-cup, Catch (b) Finger, Spin
(c) Cheetah, Run (d) Walker, Run

Fig. 4. Four experimental control tasks from the DeepMind Control Suite [17]. In all tasks, only third-person views (i.e., $64 \times 64 \times 3$ pixels) are input into the MPC-policy as observations $\boldsymbol{x}$.

PlaNet [4]. To conduct this experiment, we consider the control tasks of the DeepMind control suite [17], as shown in Fig. 4. The four difficult domains also considered in PlaNet's paper [4] are selected. Concerning the task of the "Walker" domain, a more difficult task "Run" is introduced instead of the "Walk" task originally used in [4]. In addition, for several domains, their configurations are modified so as to make training more difficult[3]. Figure 5 represents the experimental results. It can be seen that PlaNet-Bayes consistently achieves better asymptotic performance and/or faster training compared with those of PlaNet.

[1]It should be noted that the heuristics discussed in [12] is introduced at $\ell 11$–12. Let us describe the optimality likelihood as $p(\mathcal{O}_{\geq t}|\boldsymbol{z}_{\geq t}) \coloneqq f^{\mathrm{opt}}(R(\boldsymbol{z}))$, where $f^{\mathrm{opt}}(\cdot)$ is a monotonic increasing function and $R(\boldsymbol{z})$ is a trajectory reward function. The heuristics use $\mathcal{W}'(\boldsymbol{a}_{\geq t}) \coloneqq f^{\mathrm{opt}}(\mathbb{E}[R(\boldsymbol{z})])$ instead of $\mathcal{W}(\boldsymbol{a}_{\geq t}) \coloneqq \mathbb{E}[f^{\mathrm{opt}}(R(\boldsymbol{z}))]$. It has been experimentally observed that these heuristics demonstrates higher optimization performance [9], [12].

[2]https://github.com/google-research/planet

[3]For all domains except for the "Ball-in-cup", we modified the control range (i.e., torques) from $[-1, 1]$ to $[-3, 3]$, expanding action-state-spaces and emphasizing uncertainties in the posteriors. Without this, the optimal actions may often take clipped values, i.e., $\boldsymbol{a} \in \{-\mathbf{1}, \mathbf{1}\}$, meaning that the orignal tasks require rather discrete control and not continuous one. Further, in the "Cheetah" and "Walker" domains, we removed the upper limit of the reward functions.

TABLE I

THE RESULTS OF THE ABLATION STUDY IN TERMS OF MEANS AND STANDARD DEVIATION OF EPISODE REWARDS OVER 4 SEEDS AND LAST 10 TRIALS.

Method			Task			
	Model Uncertainty	Action Uncertainty	Ball-in-cup	Finger	Cheetah	Walker
PlaNet-Bayes (Ours)	✓	✓	$\mathbf{842 \pm 187}$	$\mathbf{818 \pm 54}$	$\mathbf{1738 \pm 300}$	$\mathbf{1099 \pm 153}$
	✓	-	797 ± 219	557 ± 296	1577 ± 329	1031 ± 167
	-	✓	515 ± 431	304 ± 260	1101 ± 280	896 ± 99
PlaNet [4]	-	-	719 ± 325	289 ± 284	1282 ± 295	845 ± 131

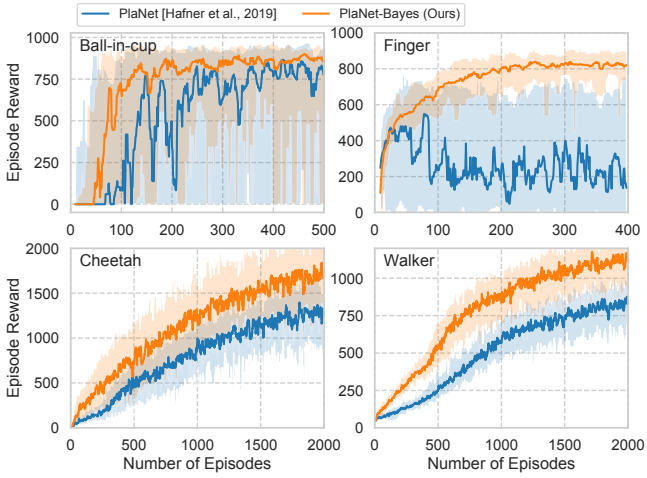


Fig. 5. Comparison between PlaNet-Bayes and the baseline PlaNet. The lines represent the medians, and the shaded areas depict the percentiles 5 to 95 over 4 seeds and 10 trajectories. The discrepancy in the performance of PlaNet with respect to [4] is caused by our task modification[3].

B. Ablation Study

This experiment is conducted to analyze how the major components of PlaNet-Bayes (model and action uncertainty) contribute to the overall improvement. Here, variants of the proposed have been prepared: either types of uncertainty are invalidated by setting either E or M equal to 1. Table I summarizes the results of the performed ablation study. Considering only model uncertainty leads to an improvement in the performance compared with PlaNet, while focusing on action uncertainty does not have any positive influence on the performance. This is because action uncertainty is rather unimodal considered alone without multimodal model uncertainty modeled by a neural network ensemble. However, when they are introduced together, multimodality in optimal actions is emphasized, and therefore, the potential of PaETS is fully exploited, achieving the best scores.

C. Video Predictions

Fig. 6 exemplifies video prediction results by PlaNet and PlaNet-Bayes, in which PlaNet-Bayes generates diverse trajectory predictions by utilizing multiple models. This uncertainty-aware behavior makes the MPC-policy avoid overfitting to unreliable predictions, and encourage active exploration in state-action spaces. In addition, PaETS makes the policy more active by exploiting various plans derived from the diverse predictions.

V. Related Work

Recently, MBRL applications to partially observable environments have been attracting great attention; namely this question was addressed in the studies [28]–[33]. In [28], the authors utilized a general VAE objective to train the latent space models and then optimized linear policies by using CMA-ES [21]. The time-series objective similar to Eq. 2 was introduced in [29], [31], and the learned models were applied to policy optimization by using Soft Actor-Critic [34]. In [33] applies PlaNet perform to imitation learning by employing adversarial training. Although most of these studies were based on autoencoding variational Bayes, posterior inference of models was not considered. In [35], there was an attempt to extend PlaNet to be Bayesian by employing variational inference of the posterior $p(\theta|\mathcal{D})$. However, as experimentally suggested in [12], a tractable variational distribution $q(\theta)$, i.e., Gaussian, is less expressive than the neural network ensemble to capture multimodality. In addition, the experiment presented in [35], which was conducted using the "Cheetah" domain, did not the achieve asymptotic performance improvement. Regarding uncertainty-aware planning, the research works mentioned above did not consider the action uncertainty. To the best of our knowledge, PlaNet-Bayes is the first attempt to incorporate uncertainty and demonstrate its efficiency by achieving better performance compared with the state-of-the-art latent MPC method: PlaNet.

VI. Conclusion

In the present paper, we proposed PlaNet-Bayes, a Bayesian extension of the state-of-the-art MBRL method for partially observable environments PlaNet. Bayesian inference of the model posterior $p(\theta|\mathcal{D})$ was introduced such as to be realized by the ensemble of latent dynamics models. The other Bayesian concept was employed for inference of the optimal trajectory posterior, which allowed formulating VI-MPC and PaETS for latent planning to successfully incorporate the multimodal uncertainty in optimal actions. By exploiting both considered uncertainty-aware strategies, PlaNet-Bayes was able to improve asymptotic performance compared to with that of baseline PlaNet.

The approaches proposed in this paper are also applicable to a variety of control methods for partially observable environments. For example, the ensemble scheme of the latent state-space model could be used to improve policy-oriented MBRL performance [28], [29], [31]. On the other hand, by introducing a categorical distribution (or a mixture model of

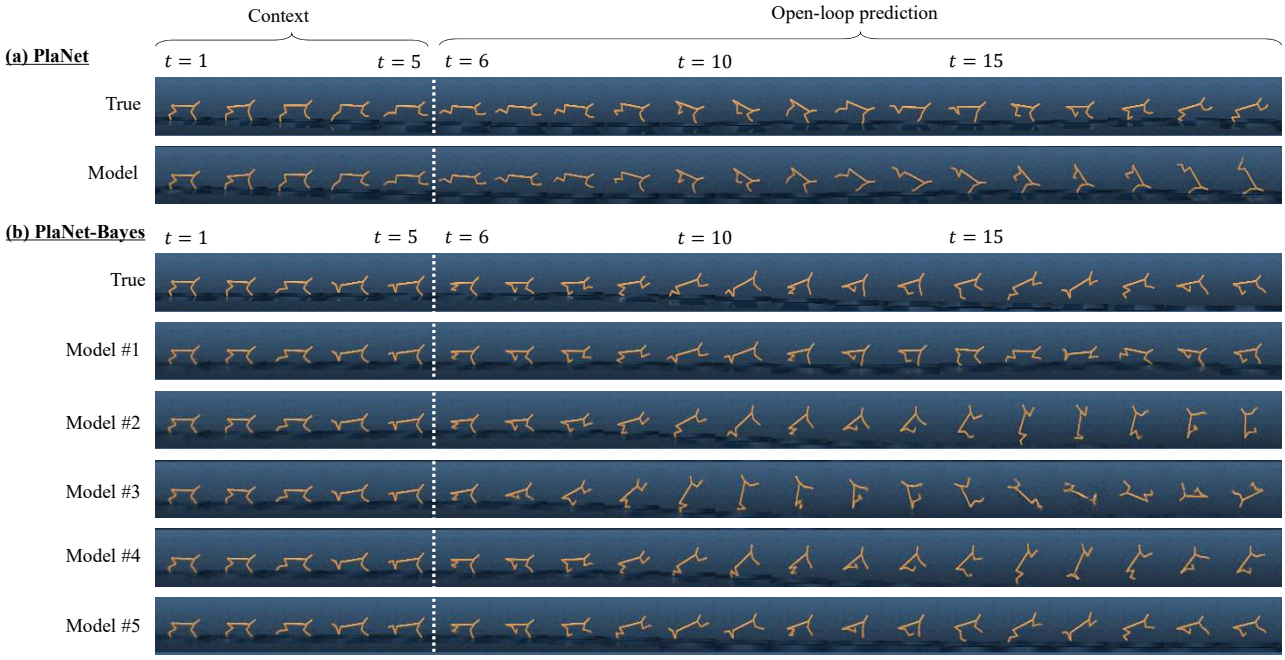


Fig. 6. Open-loop video predictions. The images $t \leq 5$ show reconstructed context frames and the remaining images are generated open-loop. Even though common inputs are fed into PlaNet-Bayes' multiple models (neural network ensemble), they output various trajectories.

it) as a variational distribution q, latent planning for discrete control tasks, including video- and board-games [32], [36], [37], is possible.

Future research directions will be related to experiments on latent MPC in real systems. Specifically in the case of industrial systems, deep understanding of the learned models and latent space is essential to obtain information about the agent's decision-making process. To represent the explainable latent space, we will consider adopting disentangling approaches [38], [39] as a promising research direction. The proposed model ensemble approach may also contribute to analysis and enhancement of the decision-making, as the confidence of trajectory prediction can be quantified by using multiple prediction outputs similarly as in [40], which suggests what the new data can be collected to improve the confidence by reducing the uncertainty.

REFERENCES

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

[2] 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*, vol. 24, no. 8, pp. 2009–2021, 2000.

[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.*, vol. 8, no. 1, pp. 16–31, 2014.

[4] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *International Conference on Machine Learning (ICML)*, 2019.

[5] M. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *International Conference on Machine Learning (ICML)*, 2011.

[6] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *International Conference on Machine Learning (ICML)*, 2016.

[7] Y. Gal, J. Hron, and A. Kendall, "Concrete dropout," in *Neural Information Processing Systems*, 2017.

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

[9] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Neural Information Processing Systems*, 2018.

[10] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, "Model-ensemble trust-region policy optimization," in *International Conference on Learning Representations (ICLR)*, 2018.

[11] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel, "Model-based reinforcement learning via meta-policy optimization," in *Conference on Robot Learning (CoRL)*, 2018.

[12] M. Okada and T. Taniguchi, "Variational inference MPC for bayesian model-based reinforcement learning," in *Conference on Robot Learning (CoRL)*, 2019.

[13] A. Nagabandi, K. Konoglie, S. Levine, and V. Kumar, "Deep dynamics models for learning dexterous manipulation," in *Conference on Robot Learning (CoRL)*, 2019.

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

[15] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations (ICLR)*, 2014.

[16] 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.

[17] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, *et al.*, "DeepMind control suite," *arXiv preprint arXiv:1801.00690*, 2018.

[18] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[19] M. Okada, L. Rigazio, and T. Aoshima, "Path integral networks: End-to-end differentiable optimal control," *arXiv preprint arXiv:1706.09597*, 2017.

[20] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *International Conference on Robotics and Automation (ICRA)*, 2016.

[21] 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*, vol. 11, no. 1, pp. 1–18, 2003.
[22] S. Goschin, A. Weinstein, and M. Littman, "The cross-entropy method optimizes for quantiles," in *International Conference on Machine Learning (ICML)*, 2013.
[23] E. Daxberger and J. M. Hernández-Lobato, "Bayesian variational autoencoders for unsupervised out-of-distribution detection," *arXiv preprint arXiv:1912.05651*, 2019.
[24] S. Bubeck *et al.*, *Convex optimization: Algorithms and complexity*, vol. 8, ch. 4. Now Publishers, Inc., 2015.
[25] M. Okada and T. Taniguchi, "Acceleration of gradient-based path integral method for efficient optimal and inverse optimal control," in *International Conference on Robotics and Automation (ICRA)*, 2018.
[26] S. Thrun, *Probabilistic robotics*, vol. 45, ch. 5, pp. 52–57. ACM New York, NY, USA, 2002.
[27] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.
[28] D. Ha and J. Schmidhuber, "World models," *arXiv preprint arXiv:1803.10122*, 2018.
[29] A. X. Lee, A. Nagabandi, P. Abbeel, and S. Levine, "Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model," *arXiv preprint arXiv:1907.00953*, 2019.
[30] M. Igl, L. Zintgraf, T. A. Le, F. Wood, and S. Whiteson, "Deep variational reinforcement learning for POMDPs," *arXiv preprint arXiv:1806.02426*, 2018.
[31] D. Han, K. Doya, and J. Tani, "Variational recurrent models for solving partially observable control tasks," in *International Conference on Learning Representations (ICLR)*, 2020.
[32] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, *et al.*, "Model-based reinforcement learning for atari," *arXiv preprint arXiv:1903.00374*, 2019.
[33] R. Okumura, M. Okada, and T. Taniguchi, "Domain-adversarial and -conditional state space model for imitation learning," *arXiv preprint arXiv:2001.11628*, 2020.
[34] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning (ICML)*, 2018.
[35] D. Tran, M. Dusenberry, M. van der Wilk, and D. Hafner, "Bayesian layers: A module for neural network uncertainty," in *Neural Information Processing Systems*, 2019.
[36] D. Silver, A. Huang, C. J. Maddison, A. Guez, *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, 2016.
[37] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, *et al.*, "Mastering Atari, go, chess and shogi by planning with a learned model," *arXiv preprint arXiv:1911.08265*, 2019.
[38] V. Thomas, E. Bengio, W. Fedus, J. Pondard, *et al.*, "Disentangling the independently controllable factors of variation by interacting with the world," *NeurIPS Learning Disentangling Representations Workshop*, 2018.
[39] Y. Sawada, "Disentangling controllable and uncontrollable factors of variation by interacting with the world," in *NeurIPS Deep Reinforcement Learning Workshop*, 2018.
[40] W. H. Beluch, T. Genewein, A. Nürnberger, and J. M. Köhler, "The power of ensembles for active learning in image classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

APPENDIX I
COMPLETE DEFINITION OF PAETS

$$w_k^{(j+1)} \leftarrow \frac{\mathcal{W}(\boldsymbol{a}_k)^{\frac{1}{\lambda}} \cdot (q^{(j)}(\boldsymbol{a}_k))^{-\kappa}}{\sum_{k'=1}^{K} \mathcal{W}(\boldsymbol{a}_{k'})^{\frac{1}{\lambda}} \cdot (q^{(j)}(\boldsymbol{a}_{k'}))^{-\kappa}} \tag{19}$$

$$\eta_m(\boldsymbol{a}_k) \coloneqq \frac{\pi_m^{(j)} \mathcal{N}(\boldsymbol{a}_k; \boldsymbol{\mu}_m^{(j)}, \boldsymbol{\sigma}_m^{(j)})}{\sum_{m'=1}^{M} \pi_{m'}^{(j)} \mathcal{N}(\boldsymbol{a}_k; \boldsymbol{\mu}_{m'}^{(j)}, \boldsymbol{\sigma}_{m'}^{(j)})} \tag{20}$$

$$\omega_{m,k}^{(j+1)} \coloneqq \eta_m(\boldsymbol{a}_k) w_k^{(j+1)} \Big/ \underbrace{\sum_{k'=1}^{K} \eta_m(\boldsymbol{a}_{k'}) w_{k'}^{(j+1)}}_{\coloneqq N_m} \tag{21}$$

$$\boldsymbol{\mu}_m^{(j+1)} \leftarrow \sum_{k=1}^{K} \omega_{m,k}^{(j+1)} \boldsymbol{a}_k \tag{22}$$

$$\boldsymbol{\sigma}_m^{(j+1)} \leftarrow \sum_{k=1}^{K} \omega_{m,k}^{(j+1)} (\boldsymbol{a}_k - \boldsymbol{\mu}_m^{(j+1)})^2 \tag{23}$$

$$\pi_m^{(j+1)} \leftarrow N_m \Big/ \sum_{m'=1}^{M} N_{m'}. \tag{24}$$