Sequential Estimation of Gaussian Process-based Deep State-Space Models

Yuhao Liu, Marzieh Ajirak, and Petar M. Djurić, *Fellow, IEEE*

***Abstract*—We consider the problem of sequential estimation of the unknowns of state-space and deep state-space models that include estimation of functions and latent processes of the models. The proposed approach relies on Gaussian and deep Gaussian processes that are implemented via random feature-based Gaussian processes. In these models, we have two sets of unknowns, highly nonlinear unknowns (the values of the latent processes) and conditionally linear unknowns (the constant parameters of the random feature-based Gaussian processes). We present a method based on particle filtering where the parameters of the random feature-based Gaussian processes are integrated out in obtaining the predictive density of the states and do not need particles. We also propose an ensemble version of the method, with each member of the ensemble having its own set of features. With several experiments, we show that the method can track the latent processes up to a scale and rotation.**

***Index Terms*—deep state-space models, deep Gaussian processes, sparse Gaussian processes, random features, ensembles, particle filtering, Bayesian linear regression**

I. Introduction

In the last decade, the field of machine learning has seen an exceptional surge and unthinkable accomplishments [1], [2]. One might argue that the main reason behind its major advances has been the much improved capabilities of deep neural networks over classical machine learning methods. Enabled by their structures of multiple processing layers, deep neural networks can learn representations of data at various levels of abstraction [3], [4], [5]. An area of machine learning that undergoes continued growth is Gaussian processes (GPs), which are now routinely employed in solving hard machine learning problems. The reason for this is that they provide a principled, practical, and probabilistic approach to learning [6]. Further, they are flexible, non-parametric, and computationally rather simple. They are used within a Bayesian framework that often leads to powerful methods which also offer valid estimates of uncertainties in predictions and generic model selection procedures [7]. Their main drawback of computational scaling has recently been alleviated by the introduction of generic sparse approximations [8], [9], [10].

GPs have also been used for building dynamical models [11]. Because of their beneficial properties, including bias-variance trade-off and their Bayesian framework, they, too, have become a tool for system identification [12]. The GP-based state-space models (GP-SSMs) describe dynamical systems, where one GP models a state process [13] and another GP models the function between the states and the observations [11].

In the literature, there have been various approaches to inference of GP-SSMs. For example, [14] and [15] discussed the combination of GP inference with various filters such as particle filters, extended Kalman filters, and unscented particle filters. Based on the reported results, the particle filters were generally the most accurate. However, the estimation of GPs in [14] and [15] requires the inversion of kernel matrices, which needs cubic time complexity. A computationally efficient way of GP-based inference was researched in [16], and it is based on approximating the GPs in feature spaces with numerous basis functions. The authors used particle Gibbs samplers for all the unknowns. In other words, they did not only sample the particles of the latent states but also the weight vectors and the hyperparameters, hence increasing the computational burden. Another family of efficient estimation of GP-SSMs is based on variational inference. In [17], [18], and [19], different evidence lower bounds (ELBOs) were designed and then optimized. These methods, however, are not sequential or online. In our work we adopted PF for estimating the hidden processes because this methodology is sequential in nature and has the capacity to perform estimation in highly nonlinear and nonstationary settings with any computable probability distributions. PF has also been used in a framework where the state-transition function of a model is parameterized using reproducing kernels [20]. Our approach in this paper can be adapted to other types of filters.

If the functions are described by deep mappings such as deep GPs, the resulting model is referred to as a GP-based deep state-space model (GP-DSSM) [21]. The analytical filters mentioned above are still applicable in deep state-space models (DSSMs). Solutions to DSSMs can be based on Rao-Blackwellized particle filters [22], [23] and mixture Kalman filters [24]. A subclass of DSSMs can be built by extending variational autoencoders (VAEs) as in [25]. The building blocks for these models are recurrent neural networks (RNNs) and VAEs.

Recently, methods for probabilistic forecasting of time series based on RNNs have been proposed [26]. The objective was to learn complex patterns from raw data by an RNN combined with a parameterized per-time-series linear state-space model. Additional efforts with similar objectives and methodologies were reported in [27]. In [28], a global-local method based on deep factor models with random effects was explored. DSSMs were also used to construct deep GPs by hierarchically putting transformed GP priors on the length scales and magnitudes of the next level of GPs in the hierarchy [29]. All these methods are different from the ones we propose here.

One way of broadening the function space of a GP is by introducing an ensemble of GPs [30], [31], [32], [33]. Each GP may rely on all or on a subset of training samples and may use a unique kernel to make predictions. Ensembles of GPs have also been used for combining global approximants with local GPs [9], [34]. In [35], an ensemble of GPs was used for online interactive learning.

We address the problem of constructing dynamic deep probabilistic latent variable models. The underlying idea is that, unlike standard state-space models, we work with DSSMs, where the variables in the intermediate layers are independently conditioned on the states from the deeper layers, and the dynamics are generated by the process from the deepest layer, the root process. An important task of inference is the estimation of the unknowns of the model, which include the underlying parameters of the GPs and the state (latent) processes of the model.

The contributions of the paper are as follows:

- a novel kernel-based method that identifies non-linear state-space systems without any information about the functions that govern the latent and observation processes,
- extension of the state-space models to deep structures to improve the model capacity and reveal more information about the studied phenomena, and
- ensemble learning to reduce the variances of the estimates of the latent processes and the predictions of the observations.

II. Background

In this section, for a self-sustained presentation, we provide some background on the methodologies that are the main ingredients of the proposed solutions in this paper.

A. Gaussian Processes

A GP, written as $\mathcal{GP}\left(m(\cdot), \kappa(\cdot,\cdot|\boldsymbol{\lambda})\right)$, is, in essence, a distribution over functions, where $m(\cdot)$ is a mean function, $\kappa(\cdot,\cdot)$ is a kernel or covariance function, and $\boldsymbol{\lambda}$ is a vector of hyperparameters that parameterize the kernel. To simplify the notation, we express a GP as $\mathcal{GP}(m,\kappa)$ or as $\mathcal{GP}(m,\kappa(\boldsymbol{\lambda}))$, if $\boldsymbol{\lambda}$ is emphasized. For any set of inputs $\mathbf{X} = [\mathbf{x}_j]_{j=1}^J := [\mathbf{x}_1,\ldots,\mathbf{x}_J]^\top$ in the domain of a real-valued function $f \sim \mathcal{GP}(m,\kappa)$, the function values $\mathbf{f} = [f(\mathbf{x}_j)]_{j=1}^J$ are Gaussian distributed, i.e.,

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}|\mathbf{m}_\mathbf{X}, \mathbf{K}_{\mathbf{XX}}), \tag{1}$$

where $\mathbf{m}_\mathbf{X} = [m(\mathbf{x}_j)]_{j=1}^J$ is the mean and $\mathbf{K}_{\mathbf{XX}} := \kappa(\mathbf{X},\mathbf{X}|\boldsymbol{\lambda}) = [\kappa(\mathbf{x}_i,\mathbf{x}_j)]_{i,j}$. Given the observations $\mathbf{f}$ on $\mathbf{X}$, the predictive distribution of $\mathbf{f}^*$ at new inputs $\mathbf{X}^*$ is given by [6]

$$p(\mathbf{f}^*|\mathbf{X}^*,\mathbf{f},\mathbf{X}) = \mathcal{N}(\mathbf{f}^*|\boldsymbol{\mu}^*,\boldsymbol{\Sigma}^*), \tag{2}$$

with a predictive mean and covariance obtained by

$$\begin{aligned}\boldsymbol{\mu}^* &= \mathbf{m}_{\mathbf{X}^*} + \mathbf{K}_{\mathbf{X}^*\mathbf{X}}\mathbf{K}_{\mathbf{XX}}^{-1}(\mathbf{f}-\mathbf{m}_\mathbf{X}),\\ \boldsymbol{\Sigma}^* &= \mathbf{K}_{\mathbf{X}^*\mathbf{X}^*} - \mathbf{K}_{\mathbf{X}^*\mathbf{X}}\mathbf{K}_{\mathbf{XX}}^{-1}\mathbf{K}_{\mathbf{X}\mathbf{X}^*}.\end{aligned} \tag{3}$$

B. Random Feature-Based Gaussian Processes

GPs do not scale up well with N, the number of input-output pairs. We observe that in (3), one has to invert the $N \times N$ matrix $\mathbf{K}_{\mathbf{XX}}$, which for large values of N becomes an issue. To ameliorate the problem, we resort to approximations by exploiting the concept of sparsity.

Compared with approximations in a function space, a GP with a shift-invariant kernel has another way of approximation, one that focuses on a feature space [36]. By utilizing feature spaces, the computations do not require matrix decompositions but only matrix multiplications. The vector of random features is comprised of trigonometric functions that are defined by

$$\begin{aligned}\boldsymbol{\phi}(\mathbf{x}) = \frac{1}{\sqrt{J}}[&\sin(\mathbf{x}^\top\boldsymbol{\omega}^1),\cos(\mathbf{x}^\top\boldsymbol{\omega}^1),\ldots,\\ &\sin(\mathbf{x}^\top\boldsymbol{\omega}^J),\cos(\mathbf{x}^\top\boldsymbol{\omega}^J)]^\top,\end{aligned} \tag{4}$$

where $\boldsymbol{\Omega} = \{\boldsymbol{\omega}^j\}_{j=1}^J$ is a set of samples randomly drawn from the power spectral density of the kernel of the GP. Then the kernel function $k(\mathbf{x},\mathbf{x}')$ can be approximated by $\boldsymbol{\phi}(\mathbf{x})^\top\boldsymbol{\phi}(\mathbf{x}')$ if the kernel is shift-invariant. It brings a type of GP approximation according to

$$f \approx \boldsymbol{\phi}(\mathbf{x})^\top\boldsymbol{\theta}, \tag{5}$$

where $\boldsymbol{\theta}$ are parameters of the approximating model.

C. Bayesian Linear Regression

In view of the model given by (5), we provide a brief review of Bayesian linear regression. Consider the following model:

$$y = \boldsymbol{\phi}^\top\boldsymbol{\theta} + \epsilon, \tag{6}$$

where y is a scalar observation, ϵ is a zero-mean Gaussian random noise, i.e., $\epsilon \sim \mathcal{N}(0,\sigma^2)$, with σ^2 being unknown, $\boldsymbol{\phi} \in \mathbb{R}^{d_\theta\times 1}$ is a known feature vector, and $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$ is an unknown parameter vector. We assume that $\boldsymbol{\theta}$ and σ^2 have a joint prior given by the multivariate normal–inverted Gamma distribution, i.e.,

$$p(\boldsymbol{\theta},\sigma^2) \propto \frac{1}{\sigma^{a_0+1}} e^{-\frac{1}{2\sigma^2}\left(b_0+(\boldsymbol{\theta}-\boldsymbol{\theta}_0)^\top\boldsymbol{\Sigma}_0^{-1}(\boldsymbol{\theta}-\boldsymbol{\theta}_0)\right)}, \tag{7}$$

where a_0, b_0, $\boldsymbol{\theta}_0$, and $\boldsymbol{\Sigma}_0$ are parameters of the prior probability density function (pdf), and where $a_0 > d_\theta$ and $b_0 > 0$. One can show that the predictive distribution of y is given by a Student's t-distribution [37], that is,

$$p(y|\boldsymbol{\phi},a_0,b_0,\boldsymbol{\theta}_0,\boldsymbol{\Sigma}_0) \propto \left(1+\frac{1}{\varphi_1}(y-\boldsymbol{\phi}^\top\boldsymbol{\theta}_0)^2\right)^{-\frac{\nu_1+1}{2}}, \tag{8}$$

where

$$\nu_1 = a_0 - d_\theta, \tag{9}$$

$$\varphi_1 = \frac{b_0}{1-\boldsymbol{\phi}^\top\boldsymbol{\Sigma}_1\boldsymbol{\phi}}, \tag{10}$$

$$\boldsymbol{\Sigma}_1 = \left(\boldsymbol{\Sigma}_0^{-1} + \boldsymbol{\phi}\boldsymbol{\phi}^\top\right)^{-1}. \tag{11}$$

Thus, for the linear model in (6), when the prior of $\boldsymbol{\theta}$ and σ^2 is given by (7), we have an analytical expression for the predictive distribution of y.

For the posterior of $\boldsymbol{\theta}$ and σ^2 we have

$$p(\boldsymbol{\theta}, \sigma^2|y, \boldsymbol{\phi}, a_0, b_0, \boldsymbol{\theta}_0, \boldsymbol{\Sigma}_0) \propto \frac{1}{\sigma^{a_1+1}} e^{-\frac{1}{2\sigma^2}\left(b_1+(\boldsymbol{\theta}-\widehat{\boldsymbol{\theta}}_1)^\top \boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\theta}-\widehat{\boldsymbol{\theta}}_1)\right)}, \quad (12)$$

where

$$a_1 = a_0 + 1, \quad (13)$$

$$b_1 = b_0 + y^2 + \boldsymbol{\theta}_0^\top \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\theta}_0 - \widehat{\boldsymbol{\theta}}_1^\top \boldsymbol{\Sigma}_1^{-1}\widehat{\boldsymbol{\theta}}_1, \quad (14)$$

$$\widehat{\boldsymbol{\theta}}_1 = \boldsymbol{\Sigma}_1\left(\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\theta}_0 + \boldsymbol{\phi} y\right). \quad (15)$$

Clearly, the posterior pdf is also a multivariate normal–inverse Gamma pdf with parameters $a_1, b_1, \boldsymbol{\theta}_1$, and $\boldsymbol{\Sigma}_1$, which are updated from $a_0, b_0, \boldsymbol{\theta}_0$ and $\boldsymbol{\Sigma}_0$ using (13), (14), (15) and (11), respectively.

D. Particle Filtering

In the proposed approach, we will use concepts from particle filtering theory, and in this subsection, we provide the basics of it. Particle filters have the capacity to work sequentially with highly nonlinear models. In many signal processing problems, we aim at tracking a latent process $\mathbf{x}_t \in \mathbb{R}^{d_x}$ of a state-space model given by

$$\text{transition pdf}: \quad p(\mathbf{x}_t|\mathbf{x}_{t-1}), \quad (16)$$

$$\text{likelihood of } \mathbf{x}_t: \quad p(y_t|\mathbf{x}_t), \quad (17)$$

where t is a discrete-time index, and $y_t \in \mathbb{R}$ is an observation process. Typically, the main objective of PF is to obtain the filtering pdf $p(\mathbf{x}_t|y_{1:t})$ from $p(\mathbf{x}_{t-1}|y_{1:t-1})$.

In brief, particle filters approximate the pdfs of interest by discrete random measures, where the support of a pdf is given by a set of particles and where each particle is given a weight following fundamental principles. PF is implemented as follows [38], [39], [40]. Suppose that at time $t-1$ the filtering density $p(\mathbf{x}_{t-1}|y_{1:t-1})$ is approximated by

$$p^M(\mathbf{x}_{t-1}|y_{1:t-1}) = \frac{1}{M}\sum_{m=1}^{M}\delta(\mathbf{x}_{t-1} - \mathbf{x}_{t-1}^{(m)}), \quad (18)$$

where the symbol $\mathbf{x}_{t-1}^{(m)}$ represents the mth particle (sample) of $\mathbf{x}_{t-1}$, $\delta(\cdot)$ is the Dirac delta function, and M is the number of particles. Then we can obtain $p^M(\mathbf{x}_t|y_{1:t})$ from $p^M(\mathbf{x}_{t-1}|y_{1:t-1})$ by implementing three steps:

1) Generate particles $\mathbf{x}_t^{(m)}$ from the predictive pdf of $\mathbf{x}_t$, i.e.,

$$\mathbf{x}_t^{(m)} \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(m)}). \quad (19)$$

2) Compute the weights of the particles $\mathbf{x}_t^{(m)}$ according to the likelihood of $\mathbf{x}_t$, or

$$w_t^{(m)} \propto p(y_t|\mathbf{x}_t^{(m)}), \quad (20)$$

and where

$$\sum_{m=1}^{M} w_t^{(m)} = 1. \quad (21)$$

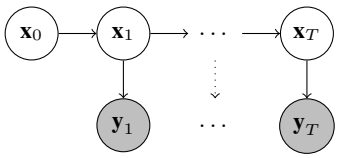


Fig. 1. A generic diagram of an SSM.

The approximation of $p(\mathbf{x}_t|y_{1:t})$ is then given by

$$p^M(\mathbf{x}_t|y_{1:t}) = \sum_{m=1}^{M} w_t^{(m)}\delta(\mathbf{x}_t - \mathbf{x}_t^{(m)}). \quad (22)$$

3) Resample the particles using their weights $w_t^{(m)}$ and construct a posterior of $\mathbf{x}_t$ with equal weights and where some of the particles are replicated [41].

III. Gaussian Process State Space Model

Now we introduce the GP-based state space model. Suppose the observation process $\mathbf{y}_t \in \mathbb{R}^{d_y}$ is produced by a state-space model defined by

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}) + \mathbf{u}_t, \quad (23)$$

$$\mathbf{y}_t = g(\mathbf{x}_t) + \mathbf{v}_t, \quad (24)$$

where (23) represents the latent state transition equation with the state vector $\mathbf{x}_t \in \mathbb{R}^{d_x}$ at time instant t, and (24) is the observation equation with $\mathbf{y}_t \in \mathbb{R}^{d_y}$ being the vector of observations at time instant t. The symbols $\mathbf{u}_t \sim \mathcal{N}(\mathbf{0}, \sigma_u^2\mathbf{I})$ and $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \sigma_v^2\mathbf{I})$ represent Gaussian distributed errors (noises). A generic graphical representation of an SSM is shown in Fig. 1.

Next, we express the above two equations using random feature-based GPs. In that case, we write them according to

$$\mathbf{x}_t = \mathbf{H}^\top\boldsymbol{\phi}^x(\mathbf{x}_{t-1}) + \mathbf{u}_t, \quad (25)$$

$$\mathbf{y}_t = \boldsymbol{\Theta}^\top\boldsymbol{\phi}^y(\mathbf{x}_t) + \mathbf{v}_t, \quad (26)$$

where the parameters are given by the elements of the matrices $\mathbf{H} \in \mathbb{R}^{2J_x \times d_x}$, $\mathbf{H} = [\boldsymbol{\eta}^{[1]}, \boldsymbol{\eta}^{[2]}, \ldots, \boldsymbol{\eta}^{[d_x]}]$, and $\boldsymbol{\Theta} \in \mathbb{R}^{2J_y \times d_y}$, $\boldsymbol{\Theta} = [\boldsymbol{\theta}^{[1]}, \boldsymbol{\theta}^{[2]}, \ldots, \boldsymbol{\theta}^{[d_y]}]$. The parameters $\boldsymbol{\eta}^{[i]}$ and $\boldsymbol{\theta}^{[j]}$ are initialized by Gaussian priors and updated by following Bayesian rules. Thus, each dimension of $\mathbf{x}_t$ and $\mathbf{y}_t$ is modeled by its own set of parameters. Further, note that the feature vectors $\boldsymbol{\phi}^x \in \mathbb{R}^{2J_x}$ and $\boldsymbol{\phi}^y \in \mathbb{R}^{2J_y}$ in (25) and (26) are different because they are defined by different sets of samples, $\boldsymbol{\Omega}^x$ and $\boldsymbol{\Omega}^y$, respectively. To simplify the notation, we use $\boldsymbol{\phi}^x(\mathbf{x}_{t-1}) =: \boldsymbol{\phi}_{t-1}^x$ and $\boldsymbol{\phi}^y(\mathbf{x}_t) =: \boldsymbol{\phi}_t^y$. We assume that the parameter variables are all independent, i.e., the columns of $\mathbf{H}$ and $\boldsymbol{\Theta}$ are independent of the remaining columns. The noises $\mathbf{u}_t$ and $\mathbf{v}_t$ are i.i.d. zero-mean Gaussians, where $\mathbf{u}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_u)$ and $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_v)$, with $\boldsymbol{\Sigma}_u = \text{diag}(\sigma_u^{2[1]}, \sigma_u^{2[2]}, \ldots, \sigma_u^{2[d_x]})$ and $\boldsymbol{\Sigma}_v = \text{diag}(\sigma_v^{2[1]}, \sigma_v^{2[2]}, \ldots, \sigma_u^{2[d_y]})$.

The model described by (25) and (26) contains many unknowns, that is, the vector processes $\mathbf{x}_t$, $t = 1, 2, \ldots$, the parameter matrices $\mathbf{H}$ and $\boldsymbol{\Theta}$, and the noise variances $\sigma_u^{2[d]}$, $d = 1, 2, \ldots . d_x$ and $\sigma_v^{2[d]}$, $d = 1, 2, \ldots . d_y$. Conditioned on $\mathbf{x}_t$, the model is of the same form as the one in (6),

whereas conditioned on $\mathbf{H}$ and $\mathbf{\Theta}$, the model given by (25) and (26) is very nonlinear in $\mathbf{x}_t$. On account of the intractable analytical inference, we resort to PF to estimate sequentially the latent states. Given the estimated states, we update the joint distributions of $\mathbf{H}$ and $\mathbf{\Sigma}_u$ and of $\mathbf{\Theta}$ and $\mathbf{\Sigma}_v$, respectively. For these updates, we apply Bayesian linear regressions, where we use multivariate normal–inverse Gamma pdfs for the joint priors of $(\boldsymbol{\eta}^{[d]}, \sigma_u^{2[d]})$ and $(\boldsymbol{\theta}^{[d]}, \sigma_v^{2[d]})$, respectively.

Next, we explain how we implement the following:

1) the propagation of $\mathbf{x}_t$,
2) the updating of the joint posteriors of $\boldsymbol{\eta}^{(m),[d]}$ and ${\sigma_u^{(m),[d]}}^2$, for $d = 1, 2, \ldots, d_x$, $m = 1, 2, \ldots, M$,
3) the updating of the joint posteriors of $\boldsymbol{\theta}^{(m),[d]}$ and ${\sigma_v^{(m),[d]}}^2$, for $d = 1, 2, \ldots, d_y$, $m = 1, 2, \ldots, M$, and
4) the weight computation of the particles and the estimation of $\mathbf{x}_t$.

Suppose that before propagating the samples of the latent process at time $t-1$, we have M particles of $\mathbf{x}_{t-1}$, $\mathbf{x}_{t-1}^{(m)}$, $m = 1, 2, \ldots, M$. Assume also that for each stream of particles m at $t-1$ we have the joint posterior of $\boldsymbol{\eta}^{[d]}$ and $\sigma_u^{2[d]}$, which is a multivariate normal–inverted Gamma pdf with parameters $a_{t-1}^x, b_{t-1}^{(m),[d]}, \boldsymbol{\eta}_{t-1}^{(m),[d]}$ and $\mathbf{\Psi}_{t-1}^{(m),[d]}$. Further, we have the joint posterior of $\boldsymbol{\theta}^{[d]}$ and $\sigma_v^{2[d]}$, which is also a multivariate normal–inverted Gamma pdf and with parameters $a_{t-1}^y, c_{t-1}^{(m),[d]}, \boldsymbol{\theta}_{t-1}^{(m),[d]}$ and $\mathbf{\Upsilon}_{t-1}^{(m),[d]}$.

A. Propagation of the particles

We generate the elements of the particles $\mathbf{x}_t^{(m)}$, $\mathbf{x}_t^{(m),[d]}$, from respective univariate Student's t-distributions given by (see also (8))

$$p(x_t^{(m),[d]}|\mathbf{x}_{t-1}^{(m)}, \mathbf{Y}_{t-1}) \propto \left(1 + \frac{1}{\psi_t^{(m),[d]}}\left(x_t^{(m),[d]} - \alpha_t^{(m),[d]}\right)^2\right)^{-\frac{\nu_{t-1}^x+1}{2}}, \quad (27)$$

where $d = 1, 2, \ldots, d_x$, $m = 1, 2, \ldots, M$, and

$$\nu_{t-1}^x = a_{t-1}^x - 2J_x, \quad (28)$$

$$\alpha_t^{(m),[d]} = {\boldsymbol{\phi}_{t-1}^{x^{(m)}}}^\top \boldsymbol{\eta}_{t-1}^{(m),[d]}, \quad (29)$$

$$\psi_t^{(m),[d]} = \frac{b_{t-1}^{(m),[d]}}{1 - {\boldsymbol{\phi}_{t-1}^{x^{(m)}}}^\top \mathbf{\Psi}_t^{(m),[d]} \boldsymbol{\phi}_{t-1}^{x^{(m)}}}, \quad (30)$$

$$\mathbf{\Psi}_t^{(m),[d]} = \left({\mathbf{\Psi}_{t-1}^{(m),[d]}}^{-1} + \boldsymbol{\phi}_t^{x^{(m)}} {\boldsymbol{\phi}_t^{x^{(m)}}}^\top\right)^{-1}. \quad (31)$$

Thus, the propagation includes generating particles $\mathbf{x}_t$ by (27). For each dimension of $\mathbf{x}_t$, we sample M particles (thus, we have a total of Md_x particles), and they represent the support of $\mathbf{x}_t$.

B. Updating of the joint posteriors of $(\boldsymbol{\eta}^{(m),[d]}, {\sigma_u^{(m),[d]}}^2)$

The joint posterior of $(\boldsymbol{\eta}^{(m),[d]}, {\sigma_u^{(m),[d]}}^2)$ is a multivariate normal–inverted Gamma pdf with parameters $a_t^x, b_t^{(m),[d]}, \boldsymbol{\eta}_t^{(m),[d]}$, and $\mathbf{\Psi}_t^{(m),[d]}$. We update $\mathbf{\Psi}_{t-1}^{(m),[d]}$ by (31), and we find the remaining parameters recursively by

$$a_t^x = a_{t-1}^x + 1, \quad (32)$$

$$b_t^{(m),[d]} = b_{t-1}^{(m),[d]} + \left(x_t^{(m),[d]}\right)^2 + {\boldsymbol{\eta}_{t-1}^{(m),[d]}}^\top {\mathbf{\Psi}_{t-1}^{(m),[d]}}^{-1} \boldsymbol{\eta}_{t-1}^{(m),[d]} - {\boldsymbol{\eta}_t^{(m),[d]}}^\top {\mathbf{\Psi}_t^{(m),[d]}}^{-1} \boldsymbol{\eta}_t^{(m),[d]}, \quad (33)$$

$$\boldsymbol{\eta}_t^{(m),[d]} = \mathbf{\Psi}_t^{(m),[d]} \left({\mathbf{\Psi}_{t-1}^{(m),[d]}}^{-1} \boldsymbol{\eta}_{t-1}^{(m),[d]} + \boldsymbol{\phi}_t^{x^{(m)}} x_t^{(m),[d]}\right). \quad (34)$$

C. Updating of the joint posteriors of $(\boldsymbol{\theta}^{(m),[d]}, {\sigma_v^{(m),[d]}}^2)$

The proposed method also requires updating of the joint posteriors of $\boldsymbol{\theta}^{(m),[d]}$ and ${\sigma_v^{(m),[d]}}^2$ for $m = 1, 2, \ldots, M$, and $d = 1, 2, \ldots, d_y$. The joint posterior of $(\boldsymbol{\eta}^{(m),[d]}, {\sigma_u^{(m),[d]}}^2)$ is a multivariate normal–inverted Gamma pdf with parameters $a_t^y, c_t^{(m),[d]}, \boldsymbol{\eta}_t^{(m),[d]}$, and $\mathbf{\Upsilon}_t^{(m),[d]}$. Upon receiving $\mathbf{y}_t$, these parameters are updated by

$$a_t^y = a_{t-1}^y + 1, \quad (35)$$

$$c_t^{(m),[d]} = c_{t-1}^{(m),[d]} + \left(y_t^{[d]}\right)^2 + {\boldsymbol{\theta}_{t-1}^{(m),[d]}}^\top {\mathbf{\Upsilon}_{t-1}^{(m),[d]}}^{-1} \boldsymbol{\theta}_{t-1}^{(m),[d]} - {\boldsymbol{\theta}_t^{(m),[d]}}^\top {\mathbf{\Upsilon}_t^{(m),[d]}}^{-1} \boldsymbol{\theta}_t^{(m),[d]}, \quad (36)$$

$$\boldsymbol{\theta}_t^{(m),[d]} = \mathbf{\Upsilon}_t^{(m),[d]} \left({\mathbf{\Upsilon}_{t-1}^{(m),[d]}}^{-1} \boldsymbol{\theta}_{t-1}^{(m),[d]} + \boldsymbol{\phi}_t^{y^{(m)}} y_t^{[d]}\right), \quad (37)$$

$$\mathbf{\Upsilon}_t^{(m),[d]} = \left({\mathbf{\Upsilon}_{t-1}^{(m),[d]}}^{-1} + \boldsymbol{\phi}_t^{y^{(m)}} {\boldsymbol{\phi}_t^{y^{(m)}}}^\top\right)^{-1}. \quad (38)$$

D. Weight computation of particles and estimation of $\mathbf{x}_t$

We need to assign weights to each particle $\mathbf{x}_t^{(m)}$ according to the likelihood of $\mathbf{x}_t^{(m)}$. The computation proceeds according to

$$\widetilde{w}_t^{(m)} = p(\mathbf{y}_t|\mathbf{x}_t^{(m)}, \mathbf{X}_{t-1}^{(m)}, \mathbf{Y}_{t-1}), \quad (39)$$

where $p(\mathbf{y}_t|\mathbf{x}_t^{(m)}, \mathbf{X}_{t-1}^{(m)}, \mathbf{Y}_{t-1})$ is the likelihood of $\mathbf{x}_t^{(m)}$ given $\mathbf{y}_t$, $\mathbf{X}_{t-1}^{(m)}$, and $\mathbf{Y}_{t-1}$, and where $\mathbf{X}_{t-1}^{(m)}$ represents all the particles generated in the mth stream up to time instant $t-1$, $\mathbf{Y}_{t-1}$ stands for all the vector observations up to time instant $t-1$, and $\widetilde{w}_t^{(m)}$ is the non-normalized weight of $\mathbf{x}_t^{(m)}$.

We obtain the likelihood by exploiting (26), where we use the made assumption that $\mathbf{v}_t$ is Gaussian. We find that $p(\mathbf{y}_t|\mathbf{x}_t^{(m)}, \mathbf{X}_{t-1}^{(m)}, \mathbf{Y}_{t-1})$ is a product of d_y Student's t-distributions, i.e.,

$$p(\mathbf{y}_t|\mathbf{x}_t^{(m)}, \mathbf{X}_{t-1}^{(m)}, \mathbf{Y}_{t-1}) \propto \prod_{d=1}^{d_y} \left(1 + \frac{1}{\upsilon_t^{(m),[d]}}\left(y_t^{[d]} - \beta_t^{(m),[d]}\right)^2\right)^{-\frac{\nu_{t-1}^y+1}{2}}, \quad (40)$$

where $d = 1, 2, \ldots, d_y$, $m = 1, 2, \ldots, M$, and

$$\nu_{t-1}^{y} = a_{t-1}^{y} - 2J_y, \tag{41}$$

$$\beta_t^{(m),[d]} = \boldsymbol{\phi}_t^{y^{(m)}\top} \boldsymbol{\theta}_{t-1}^{(m),[d]}, \tag{42}$$

$$\upsilon_t^{(m),[d]} = \frac{c_{t-1}^{(m),[d]}}{1 - \boldsymbol{\phi}_t^{y^{(m)}\top} \boldsymbol{\Upsilon}_t^{(m),[d]} \boldsymbol{\phi}_t^{y^{(m)}}}. \tag{43}$$

Once we compute the non-normalized weights by (39), we normalize them according to

$$w_t^{(m)} = \frac{\widetilde{w}_t^{(m)}}{\sum_{k=1}^{M} \widetilde{w}_t^{(k)}}. \tag{44}$$

After normalizing the weights, the minimum mean square estimate (MMSE) of $\mathbf{x}_t$ is obtained by

$$\widehat{\mathbf{x}}_t = \sum_{m=1}^{M} w_t^{(m)} \mathbf{x}_t^{(m)}. \tag{45}$$

The approximation of the posterior $p(\mathbf{x}_t|\mathbf{Y}_t)$ is then given by

$$p^M(\mathbf{x}_t|\mathbf{Y}_t) = \sum_{m=1}^{M} w_t^{(m)} \delta\left(\mathbf{x}_t - \mathbf{x}_t^{(m)}\right). \tag{46}$$

Finally, we resample M particles $\mathbf{x}_t^{(m)}$ from $p^M(\mathbf{x}_t|\mathbf{Y}_t)$ to obtain the particles that will be used for propagation in the next time instant $t+1$.

The complete procedure is summarized by Algorithm 1. We point out that an alternative algorithm can be applied where all the particles share the same parameters $\mathbf{H}$ and $\boldsymbol{\Theta}$.

Algorithm 1: Single Sequential GP-SSM

for $m = 1$ to M **do**
- Sample $\mathbf{x}_1^{(m)} \sim p(\mathbf{x}_1)$;
- Initialize the weights of $\mathbf{x}_1^{(m)}$ as $w_1^{(m)} = 1/M$, $\forall m$;

for $t = 2$ to T **do**
- **Propagation of the states:**
- Sample $\mathbf{x}_t^{(m)}$ according to (27);
- **Updating the parameters of the joint posterior of** $(\boldsymbol{\eta}^{(m),[d]}, \sigma_u^{(m),[d]^2})$**:**
- Update $a_t^x, b_t^{(m),[d]}, \boldsymbol{\eta}_t^{(m),[d]}$, and $\boldsymbol{\Psi}_t^{(m),[d]}$ via (32), (33), (34), and (31), $\forall d$ and m;
- **Updating the parameters of the joint posterior of** $(\boldsymbol{\theta}^{(m),[d]}, \sigma_v^{(m),[d]^2})$**:**
- Update $a_t^y, c_t^{(m),[d]}, \boldsymbol{\theta}_t^{(m),[d]}$, and $\boldsymbol{\Upsilon}_t^{(m),[d]}$ via (35), (36), (37), and (38), $\forall d$ and m;
- **Weight computation and normalization:**
- Compute the weights of $\mathbf{x}_t^{(m)}$ according to (39) and normalize them by (44).
- **Estimation of the state:**
- Estimate $\mathbf{x}_t$ by (45).
- **Resampling:**
- Resample $\mathbf{x}_t^{(m)}$ based on their weights.

IV. Ensemble Learning

The use of only a single set of random samples, $\boldsymbol{\Omega}$, might not be sufficiently accurate. In order to mitigate the problem, we introduce an ensemble of different sets of $\boldsymbol{\Omega}$ and then combine the results obtained by each set. Let κ^s be a shift-invariant kernel from a known kernel dictionary $K := \{\kappa^1, ..., \kappa^S\}$. Ideally, K should be built as large as computational resources allow. We create the sets $\boldsymbol{\Omega}^s$ by sampling from the power spectral density of each kernel candidate κ^s. For estimating the latent state, we use these sets as follows. If $\boldsymbol{\Omega}^s$ is the sth set, the posterior contribution or weight of the GP based on the sth set to the estimate of the latent state at time t is $w_t^s \propto p(s|\mathbf{Y}_t)$. Then, the predictive density of $\mathbf{y}_t$ at time t is obtained from

$$\begin{aligned} p(\mathbf{y}_t|\mathbf{Y}_{t-1}) &= \sum_{s=1}^{S} p(s|\mathbf{Y}_{t-1}) p(\mathbf{y}_t|s, \mathbf{Y}_{t-1}) \\ &= \sum_{s=1}^{S} w_{t-1}^s p(\mathbf{y}_t|s, \mathbf{Y}_{t-1}), \end{aligned} \tag{47}$$

where S is the total number of sets and where the posterior weight is updated by

$$w_t^s = \frac{p(s|\mathbf{Y}_{t-1}) p(\mathbf{y}_t|s, \mathbf{Y}_{t-1})}{p(\mathbf{y}_t|\mathbf{Y}_{t-1})} \propto w_{t-1}^s p(\mathbf{y}_t|s, \mathbf{Y}_{t-1}). \tag{48}$$

A. Ensemble Estimates of the States

The ensemble estimate of the latent states is given by the mixture

$$p(\widehat{\mathbf{x}}_t|\mathbf{Y}_t) = \sum_{s=1}^{S} w_t^s p(\widehat{\mathbf{x}}_t|s, \mathbf{Y}_t). \tag{49}$$

We point out that the estimates of the latent states by random feature-based methods are identifiable up to a scale, shift, and rotation [42]. Thus, to fuse the state estimates, we have to force the estimators of all the ensemble members into the same coordinate base. To that end, we arbitrarily fix the rotation of $\mathbf{X} \in \mathbb{R}^{T \times d_x}$ by taking the singular value decomposition (SVD) of the MMSE estimate, $\widehat{\mathbf{X}} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$, and setting the new estimated $\widehat{\mathbf{X}} \in \mathbb{R}^{T \times d_x}$ as the columns of the left singular vectors $\mathbf{U}$ with d_x largest singular values. Then we mirror and rotate all the candidate latent states so that they have the same pattern. First, we set a guidance point $\tilde{\mathbf{x}}_t$ with respect to a specific time t. Then we rotate all the latent states $\tilde{\mathbf{X}}^{(s)}$ to make sure that $\tilde{\mathbf{x}}_t^{(s)}$ is "overlapped" with $\tilde{\mathbf{x}}_t$. Finally, we take the weighted average of $\tilde{\mathbf{X}}^{(s)}$ as the ensemble estimate of $\tilde{\mathbf{X}}$.

B. Keep and Drop

We use the individual estimates of the ensemble members to improve on their respective estimates. In practice, if we do not take precautionary measures, only a small portion of them would remain with significant weights. For this reason, we remove the members with small weights using the principle of resampling and replace them with members that perform much better. With replacements, we reduce the diversity of features in the ensemble but increase the number of particles that explore the spaces of the latent processes with the features

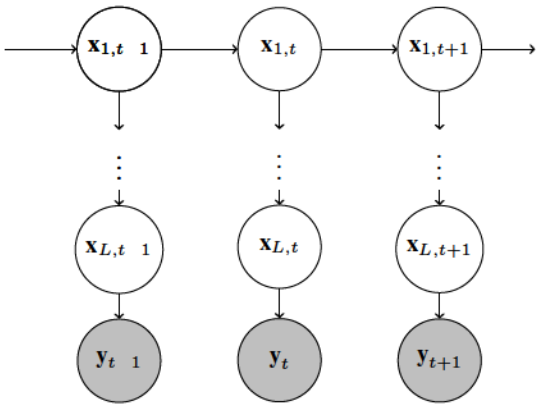


Fig. 2. A generic diagram of a deep SSM with L layers.

of the replicated members. Further, we note that candidate models need to be trained at the beginning. For this stage, we fix the weights to $w_t^s \equiv 1/S$ at the beginning until $t = \mathrm{T}_0$.

V. Gaussian Process-based Deep State-Space Models

One of the advantages of deep structures is to use one or a few simple nonlinear activation functions to improve the approximation of unknown highly nonlinear target functions. In the field of signal processing, the advantage of deep structures of state-space models is that with more hidden layers we can improve the modeling capacity of the model. Typically, the state processes of the hidden layers will be of different dimensions, and in some settings, the deep models can be justified using arguments that reflect our understanding of the phenomena we model. A generic diagram of a deep SSM with L layers is shown in Fig. 2.

Borrowing from concepts of deep learning, we introduce a Gaussian process-based deep state-space model (GP-DSSM). This model uses one simple kernel that is combined with a deep structure to approximate the unknown target kernel. Formally, we express a GP-DSSM with L hidden layers as follows:

$$\mathbf{x}_{1,t} = \mathbf{H}_1^\top \boldsymbol{\phi}_{1,t-1}^x + \mathbf{u}_{1,t}, \tag{50}$$

$$\mathbf{x}_{l,t} = \mathbf{H}_l^\top \boldsymbol{\phi}_{l-1,t}^x + \mathbf{u}_{l,t}, \quad l = 2\ldots, L, \tag{51}$$

$$\mathbf{y}_t = \boldsymbol{\Theta}^\top \boldsymbol{\phi}_t^y + \mathbf{v}_t, \tag{52}$$

where $\mathbf{x}_{l,t} \in \mathbb{R}^{d_x^l}$, $l = 1, 2, \ldots, L$ are latent processes, $\mathbf{y}_t \in \mathbb{R}^{d_y}$ is a vector of observations, $\boldsymbol{\phi}_{l,t}^x = \boldsymbol{\phi}(\mathbf{x}_{l,t})$ are the feature functions embedded with different $\boldsymbol{\Omega}_l$ for every layer, $\boldsymbol{\phi}_t^y$ has the same meaning as before, $\mathbf{H}_l$ and $\boldsymbol{\Theta}$ are parameter variables, and $\mathbf{u}_{l,t}$ and $\mathbf{v}_t$ are perturbations. We refer to the deepest latent process (defined by (50)) as the root process of the model. Here we assume that the dimensions of the latent processes are predefined. The objective of inference is to estimate all the latent processes $\mathbf{x}_{l,t}$, $l = 1, \ldots, L$ and all the parameters of the model $\mathbf{H}$ and $\boldsymbol{\Theta}_l$, $l = 1, 2, \ldots, L$,.

The inference method and procedures are very similar to the method we described for the ordinary GP-SSM.

A. Propagation of the particles in all layers

At time t, first we propagate the particles from $\mathbf{x}_{1,t-1}^{(m)}$ to $\mathbf{x}_{1,t}^{(m)}$ and then the particles of the remaining latent processes $\{\mathbf{x}_{l,t}^{(m)}\}_{l=2}^L$. In propagating these particles, we apply analogous Student's t-distributions as in (8), i.e.,

$$p(x_{l,t}^{(m),[d]} | \mathbf{X}_{t-1}^{(m)}, \mathbf{Y}_{t-1}) \propto \left(1 + \frac{1}{\psi_{l,t}^{(m),[d]}} \left(x_{l,t}^{(m),[d]} - \alpha_{l,t}^{(m),[d]}\right)^2\right)^{-\frac{\nu_{l,t-1}^x + 1}{2}}, \tag{53}$$

where $\mathbf{X}_{t-1}^{(m)}$ represents the latent states in all the layers up to time $t-1$, and where the parameters $\nu_{l,t}, \psi_{l,t}$, and $\alpha_{l,t}$ are defined similarly as in (28) – (31).

B. Updating of the joint posteriors of $(\boldsymbol{\eta}_l^{(m),[d]}, \sigma_{l,v}^{(m),[d]^2})$ and $(\boldsymbol{\theta}^{(m),[d]}, \sigma_v^{(m),[d]^2})$

These updates follow the schemes described by (32)–(34) and (35)–(38), respectively. We note that these updates can be performed in parallel once all the particles in all the layers have been propagated.

C. Weight computation of particles and estimation of the latent processes

We assign weights to the particles $\mathbf{x}_{L,t}^{(m)}$ according to the likelihoods of the particles, that is, we use

$$\widetilde{w}_{L,t}^{(m)} = p(\mathbf{y}_t | \mathbf{x}_{L,t}^{(m)}, \mathbf{X}_{t-1}^{(m)}, \mathbf{Y}_{t-1}), \tag{54}$$

where $p(\mathbf{y}_t | \mathbf{x}_{L,t}^{(m)}, \mathbf{X}_{t-1}^{(m)}, \mathbf{Y}_{t-1})$ is the likelihood of $\mathbf{x}_{L,t}^{(m)}$ given $\mathbf{y}_t$, $\mathbf{X}_{t-1}^{(m)}$, $\mathbf{Y}_{t-1}$, and $\widetilde{w}_{L,t}^{(m)}$ is the non-normalized weight of $\mathbf{x}_{L,t}^{(m)}$. The computation of this weight is carried out via a Student's t-distribution of the form as in (40) and whose parameters are from expressions analogous to (41)–(43). Upon the computation of the weights, we normalize them as per (44). Clearly, these weights directly depend on $\mathbf{x}_{L,t}^{(m)}$ only and not on the particles from the previous layers. Finally, the minimum mean square estimate (MMSE) of $\mathbf{x}_{L,t}$ is computed by (45) and the approximation of the posterior $p(\mathbf{x}_{L,t} | \mathbf{Y}_t)$ is given by (46).

The estimates of the remaining processes is carried out by first computing the weights of the particles of the corresponding processes. For example, for computing the weights of $\mathbf{x}_{L-1,t}^{(m)}$, we use

$$\widetilde{w}_{L-1,t}^{(m)} = p\left(\widehat{\mathbf{x}}_{L,t} | \mathbf{x}_{L-1,t}^{(m)}\right), \tag{55}$$

where $\widehat{\mathbf{x}}_{L,t}$ is the estimate of $\mathbf{x}_{L,t}$. From the particles $\mathbf{x}_{L-1,t}^{(m)}$ and their corresponding weights, we then compute $\widehat{\mathbf{x}}_{L-1,t}$. We continue in the same vein by estimating one process value at a time until we complete these steps with estimating the value of the root process.

Before we proceed to process the next observation, we resample the M streams using their respective weights $w_t^{(m)}$.

One approach to computing these weights ise based on the following expression:

$$\widetilde{w}_t^{(m)} = p\left(\mathbf{y}_t|\mathbf{x}_{L,t}^{(m)}, \mathbf{Y}_{t-1}\right) \prod_{l=1}^{L-1} p\left(\mathbf{x}_{l+1,t}|\mathbf{x}_{l,t}^{(m)}\right). \quad (56)$$

For the unknown $\mathbf{x}_{l+1,t}$ in this equation, we could use their respective MMSE estimates $\widehat{\mathbf{x}}_{l,t}$. Another approach is based on approximating the factors $p\left(\mathbf{x}_{l+1,t}|\mathbf{x}_{l,t}^{(m)}\right)$ in (56) with the average likelihood, that is, with

$$p\left(\mathbf{x}_{l+1,t}|\mathbf{x}_{l,t}^{(m)}\right) \approx \sum_{m'=1}^{M} w_{l+1,t}^{(m')} p\left(\mathbf{x}_{l+1,t}^{(m')}|\mathbf{x}_{l,t}^{(m)}\right), \quad (57)$$

where $w_{l+1,t}^{(m)}$ are the weights associated with the particles $x_{l+1,t}^{(m)}$. By combining equation (56) and (57), we obtain

$$\widetilde{w}_t^{(m)} \approx p\left(\mathbf{y}_t|\mathbf{x}_{L,t}^{(m)}, \mathbf{Y}_{t-1}\right) \prod_{l=1}^{L-1} \left[\sum_{m'=1}^{M} w_{l+1,t}^{(m')} p\left(\mathbf{x}_{l+1,t}^{(m')}|\mathbf{x}_{l,t}^{(m)}\right)\right]. \quad (58)$$

VI. Experiments

We tested the performance of the proposed method with several experiments. In all the experiments, we applied the ensemble method with $S = 100$ members. Specifically, the elements of the sets $\{\mathbf{\Omega}^s\}_{s=1}^{100}$ were randomly sampled from the power spectral density of RBF kernels $\kappa^s(\boldsymbol{\lambda})$ with prior length scale vectors $\boldsymbol{l}_\lambda^s$ and prior variances $\sigma_\lambda^2 = 1$, where the elements of $\boldsymbol{l}_\lambda^s$ were independently sampled from the discrete set $\{10^{-4}, 10^{-3}, \ldots, 10^3, 10^4\}$. Note that $\boldsymbol{\lambda}$ represents the hyperparameters and $\boldsymbol{\lambda} = \{\boldsymbol{l}_\lambda, \sigma_\lambda^2\}$ includes the length scale vector and the prior variances.

A. A test with $d_x = 2$ and $d_y = 1$

In the first experiment, we tested the inference of a GP-SSM when $d_x > d_y$. More specifically, we generated data from an SSM with $d_x = 2$ and $d_y = 1$ according to the following model:

$$\begin{aligned}
\text{latent layer}: \quad & x_t^{[1]} = 0.9x_{t-1}^{[1]} + 0.5\sin(x_{t-1}^{[2]}) + u_t^{[1]}, \\
& x_t^{[2]} = 0.5\cos(x_{t-1}^{[1]}) + 0.9x_{t-1}^{[2]} + u_t^{[2]}, \\
\text{observations}: \quad & y_t = 0.3\sin(x_t^{[1]}) - 0.3x_t^{[1]} + 0.2x_t^{[2]} \\
& +0.25x_t^{[1]}x_t^{[2]} + (0.05x_t^{[1]})^2 + 0.01(x_t^{[1]})^3 \\
& -0.25x_t^{[1]}/(1 + (x_t^{[2]})^2) + v_t.
\end{aligned}$$

The generated data set contained 2,000 samples with $\sigma_u^2 = \sigma_v^2 = 0.001$, and for initializing the estimation, we used $T_0 = 1,000$ samples. For drawing random vectors needed in the construction of the random features $\boldsymbol{\phi}(\mathbf{x})$ in (4), we used $J_x = J_y = 50$. For comparison purposes, all the signals were normalized from $T_0 = 1,000$ to $T = 2,000$. We emphasize again that the functions in the state and observation equations are unknown. Figure 3 shows the last 100 samples of the actual latent process $x_t^{[1]}$ in red line, its estimate $\widehat{x}_t^{[1]}$ in grey line, and the 95% confidence interval depicted by the blue region. Figure 4 exhibits the estimates of the latent states $\widehat{x}_t^{[2]}$ under 20 runs with different seeds. The results suggest that with our approach we can provide accurate estimates of the latent states and thus can capture their dynamics. Further, we can quantify the uncertainties of the estimates. Recall from Section IV-A that we use the SVD to standardize both the actual and estimated states. Figure 5 illustrates the true pairs $(x_t^{[1]}, x_t^{[2]})$ and estimated pairs $(\widehat{x}_t^{[1]}, \widehat{x}_t^{[2]})$ before applying SVD. The scaled actual and estimated states after SVD are shown in Fig. 6.

To assess the performance of our method further, in Fig. 7 we present the results of the first 100 samples, demonstrating the consistent performance of our method. It is important to note that the samples from T_0 to the end have been standardized simultaneously, ensuring a consistent standardization approach across the entire test set. Thus, the first and last 100 samples have not been separately standardized. In addition, in Fig. 8 we show the root mean square errors (RMSEs) of the estimated latent states.

Here we provide motivation and insights for using Student's t-distributions rather than Gaussian ones. In our previous work, [43], we considered the variances of the Gaussians $\boldsymbol{\sigma}^2$ as vectors that are optimized by gradient descent algorithms. However, bad initial values of variances would incur huge bias because of the risk of not converging. Therefore, we used Student's t-distributions to account for the uncertainty of the variances $\boldsymbol{\sigma}^2$. Further, the Student's t-distribution allows for a closed form formulation of the variance updates. To make a comparison between the models with Student's t and Gaussian distributions, we conducted the following experiment. We assumed that the prior information provides initial values of Gaussian variances with 0.1, while the actual variances were $\sigma_u^2 = \sigma_v^2 = 0.001$. The model with Gaussians had a much worse performance in accuracy and had increased computing time compared to the model with Student's t distributions. The results are shown in Figs. 9 and 10. The red lines show the RMSEs and MNLLs under the model with Gaussians, whereas the grey lines represent our proposed model with Student's t distributions. We reiterate that the model with Gaussians requires more time to run for the same number of samples.

B. A test with $d_x = 5$ and $d_y = 100$

In the next experiment, we tested the GP-SSM when $d_x < d_y$. We wanted to demonstrate the ability of our model to learn lower dimensional processes from high dimensional observation signals. Our generative model had $d_x = 5$ and $d_y = 100$ and was of the form

$$\begin{aligned}
\text{Latent Layer}: \quad & x_t^{[i]} = \boldsymbol{\phi}_x^\top(\mathbf{x}_{t-1})\boldsymbol{\eta}^{[i]} + u_t^{[i]}, \\
\text{Observations}: \quad & y_t^{[j]} = \boldsymbol{\phi}_y^\top(\mathbf{x}_t)\boldsymbol{\theta}^{[j]} + v_t^{[j]},
\end{aligned}$$

where $i = 1, \ldots, 5$, $j = 1, \ldots, 100$, $\boldsymbol{\phi}_x^\top(x) = [\sin(\boldsymbol{\omega}_x^\top x)\ \cos(\boldsymbol{\omega}_x^\top x)]$ and $\boldsymbol{\phi}_y^\top(x) = [\sin(\boldsymbol{\omega}_y^\top x)\ \cos(\boldsymbol{\omega}_y^\top x)]$. The elements of $\boldsymbol{\omega}_x \in \mathbb{R}^{50}$ and $\boldsymbol{\omega}_y \in \mathbb{R}^{50}$ were randomly generated from -10 to 10, and the entries of $\boldsymbol{\eta} \in \mathbb{R}^{100}$ and $\boldsymbol{\theta} \in \mathbb{R}^{100}$ were also randomly drawn from -0.01 to 0.01. The hyperparameters were set to be the same as in the above section. Figure 11 shows $\mathbf{x}_t$ and $\widehat{\mathbf{x}}_t$ of the last 100 samples. The

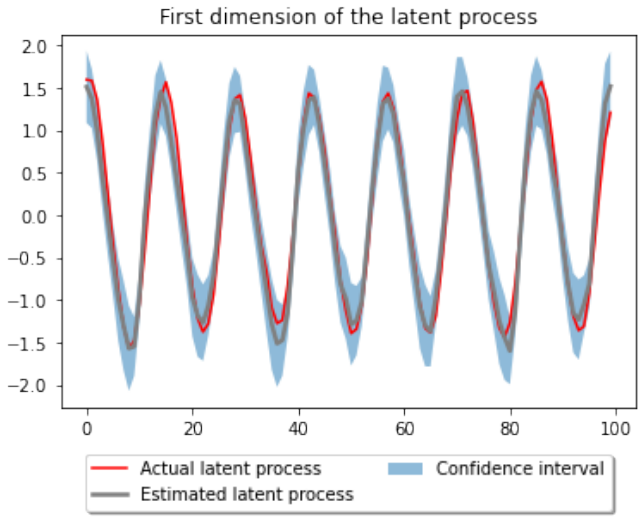


Fig. 3. Point estimates and 95% confidence region of $\widehat{x}_t^{[1]}$ and its true values $x_t^{[1]}$ for the last 100 samples.

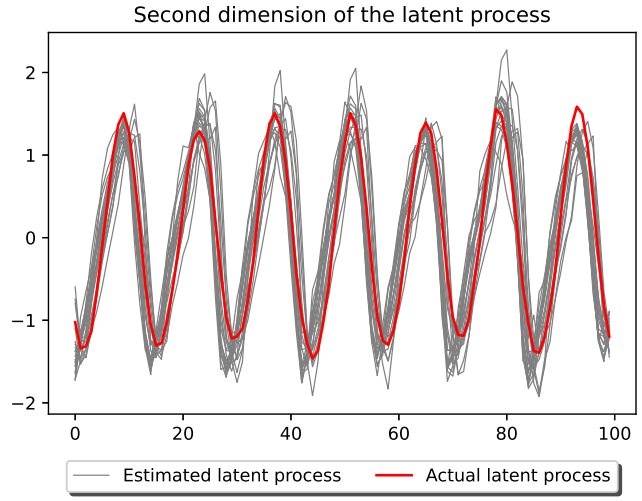


Fig. 4. Estimates of $\widehat{x}_t^{[2]}$ under 20 runs and its true values $x_t^{[2]}$.

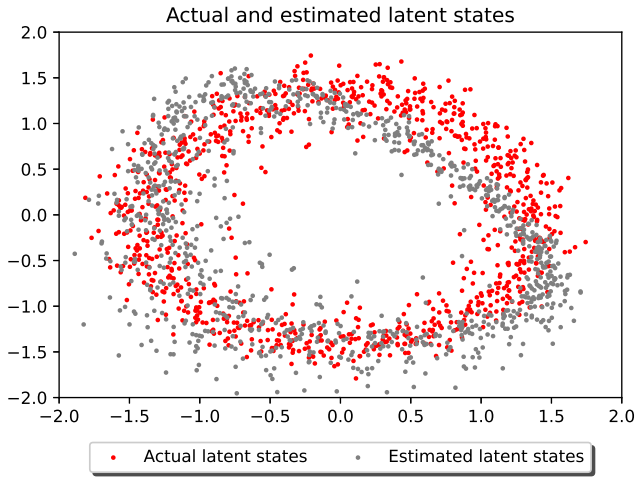


Fig. 5. Pairs of estimated ($\widehat{\mathbf{x}}_t$) and actual ($\mathbf{x}_t$) latent states before SVD.

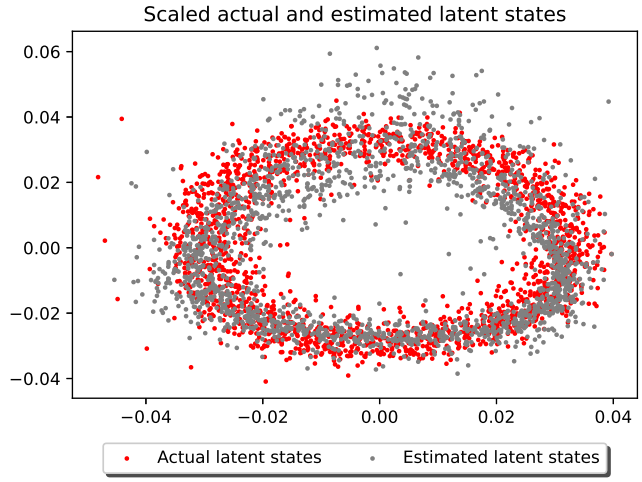


Fig. 6. Pairs of standardized $\widehat{\mathbf{x}}_t$ and $\mathbf{x}_t$ after SVD.

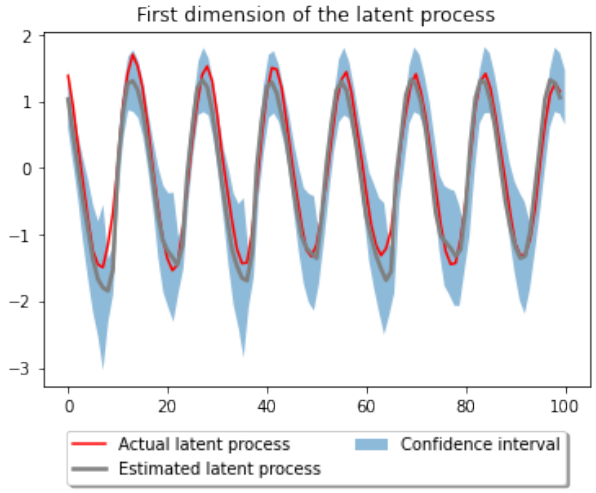


Fig. 7. Point estimates of $\widehat{x}_t^{[1]}$ with 95% confidence region and the true $x_t^{[1]}$ from T_0 to $T_0 + 100$.

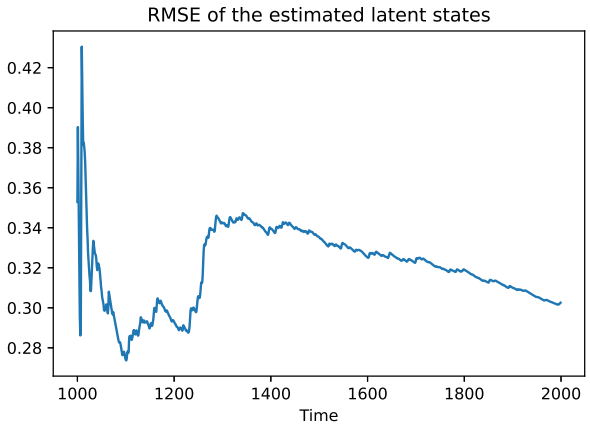


Fig. 8. RMSEs of the estimated latent states from T_0 to the end.

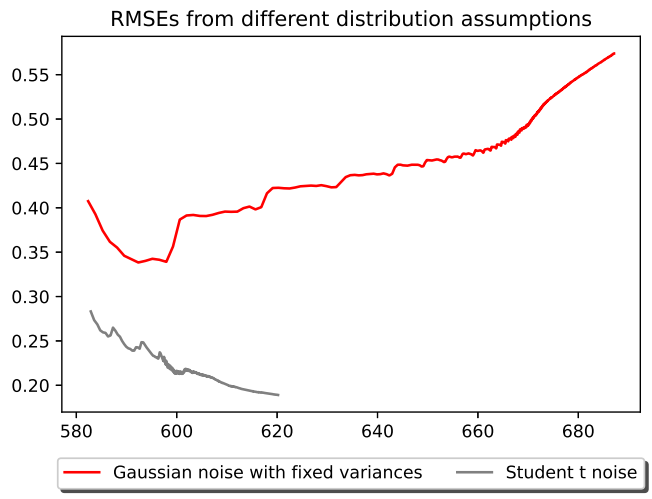


Fig. 9. RMSEs after T_0. The x-axis represents time (seconds).

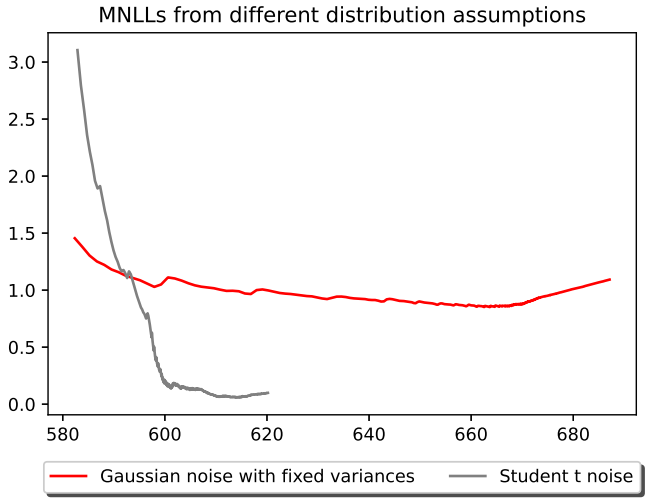


Fig. 10. MNLLs after T_0. The x-axis represents time (seconds).

results indicate that even for the signals with high frequency and dimensions, our model can adjust quickly.

C. The need for a deep model

Do we need GP-DSSMs? This experiment shows that the answer is positive, especially when the selected kernel for the GP may not have enough capacity to learn. We validated this by an experiment where $d_x = 10$ and $d_y = 1$. We generated the observation process as a GP whose kernel was a superposition of a dot-product and a Matérn kernel. The dot-product kernel was a non-stationary kernel with a hyperparameter $\sigma_{dp}^2 = 20$, and the Matérn kernel was set with hyperparameters $\nu = 1.5$ and length scales $\boldsymbol{l} = [10^{-5}, 10^{-4}, \ldots, 10^3, 10^4]$. The outputs were normalized before being used by our model. In mathematical terms, the transition and observation processes were obtained by

$$x_t^{[i]} = 0.9x_{t-1}^{[i]} + g_i(x_t^{[i+1]})/2 + u_t^{[i]}, \tag{59}$$

$$y_t = f(x_t) + v_t, \tag{60}$$

where $i = 1, \ldots, 10$, and $x_t^{[11]}$ actually denotes $x_t^{[1]}$ to simplify the notation. The function f is the GP with a dot-product kernel adding a Matérn kernel, and g_i is a sine function when i is odd while a cosine when i is even. The noises $u_t^{[i]}$ and v_t had the same variances $\sigma_u^2 = \sigma_v^2 = 0.1$. The remaining parameters were $J_x = J_y = 100$ and $M = 10^4$.

We implemented four models, from one-hidden layer to four-hidden layers. Figure 12 shows the RMSEs of the four models. The model with two-hidden layers achieves the minimum RMSEs. Note that the behavior of the RMSEs relative to the number of hidden layers is similar to a concave function, consistent with the conclusion from [44], i.e., that the RMSEs decrease and then increase with the number of hidden layers increasing. From the conclusion in [45], we might expect that deeper models would be better when we increase the number of parameters such as J and M.

D. Testing the performance of GP-DSSM

We generated data from a DSS model with two hidden layers, with $d_{x_1} = 2$, $d_{x_2} = 3$, and $d_y = 4$. The model is given by the following two layers of latent processes:

$$\begin{aligned}
\text{Layer 1}: \quad & x_{1,t}^{[1]} = 0.9x_{1,t-1}^{[1]} + 0.5\sin(x_{1,t-1}^{[1]}) + u_{1,t}^{[1]}, \\
& x_{1,t}^{[2]} = 0.5\sin(x_{1,t-1}^{[1]}) + 0.9x_{1,t-1}^{[2]}] + u_{1,t}^{[1]}, \\
\text{Layer 2}: \quad & x_{2,t}^{[1]} = 1.8\cos(x_{1,t}^{[1]}) - 0.7\sin(x_{1,t}^{[1]}) + u_{2,t}^{[1]}, \\
& x_{2,t}^{[2]} = 0.5x_{1,t}^{[1]} - 1.3\sin(x_{1,t}^{[2]}) + u_{2,t}^{[1]}, \\
& x_{2,t}^{[3]} = 2x_{1,t}^{[1]} - 0.4x_{1,t}^{[2]} + u_{2,t}^{[2]},
\end{aligned} \tag{61}$$

and four observation processes given by

$$\begin{aligned}
y_t^{[1]} &= 0.01(x_{2,t}^{[1]})^2 + 1.2x_{2,t}^{[3]} + v_t^{[1]}, \\
y_t^{[2]} &= 1.2\sin(x_{2,t}^{[1]}) - 0.5x_{2,t}^{[2]} + 0.7x_{2,t}^{[3]} + v_t^{[2]}, \\
y_t^{[3]} &= x_{2,t}^{[1]}x_{2,t}^{[2]} + v_t^{[3]}, \\
y_t^{[4]} &= 5x_{2,t}^{[2]}/(1 + x_{2,t}^{[2]^2}) + v_t^{[4]}.
\end{aligned} \tag{62}$$

This model is identical to the one from Sec. V-B of [43], except that we changed the first equation in [43]

$$x_{1,t}^{[1]} = 0.9x_{1,t-1}^{[1]} + 0.5\sin(x_{1,t-1}^{[2]}) + u_{1,t}^{[1]}, \tag{63}$$

to

$$x_{1,t}^{[1]} = 0.9x_{1,t-1}^{[1]} + 0.5\sin(x_{1,t-1}^{[1]}) + u_{1,t}^{[1]}. \tag{64}$$

We made this change to force the latent process to become less smooth, thus making it harder for estimation. The results are shown in Fig. 13. Evidently, they show that the proposed method is capable of accurately estimating all the latent processes even though the latent processes are much more jagged. Further, compared with the results in [43] which had persistent lags in the estimated processes, the results from the method presented here showed almost no lags.

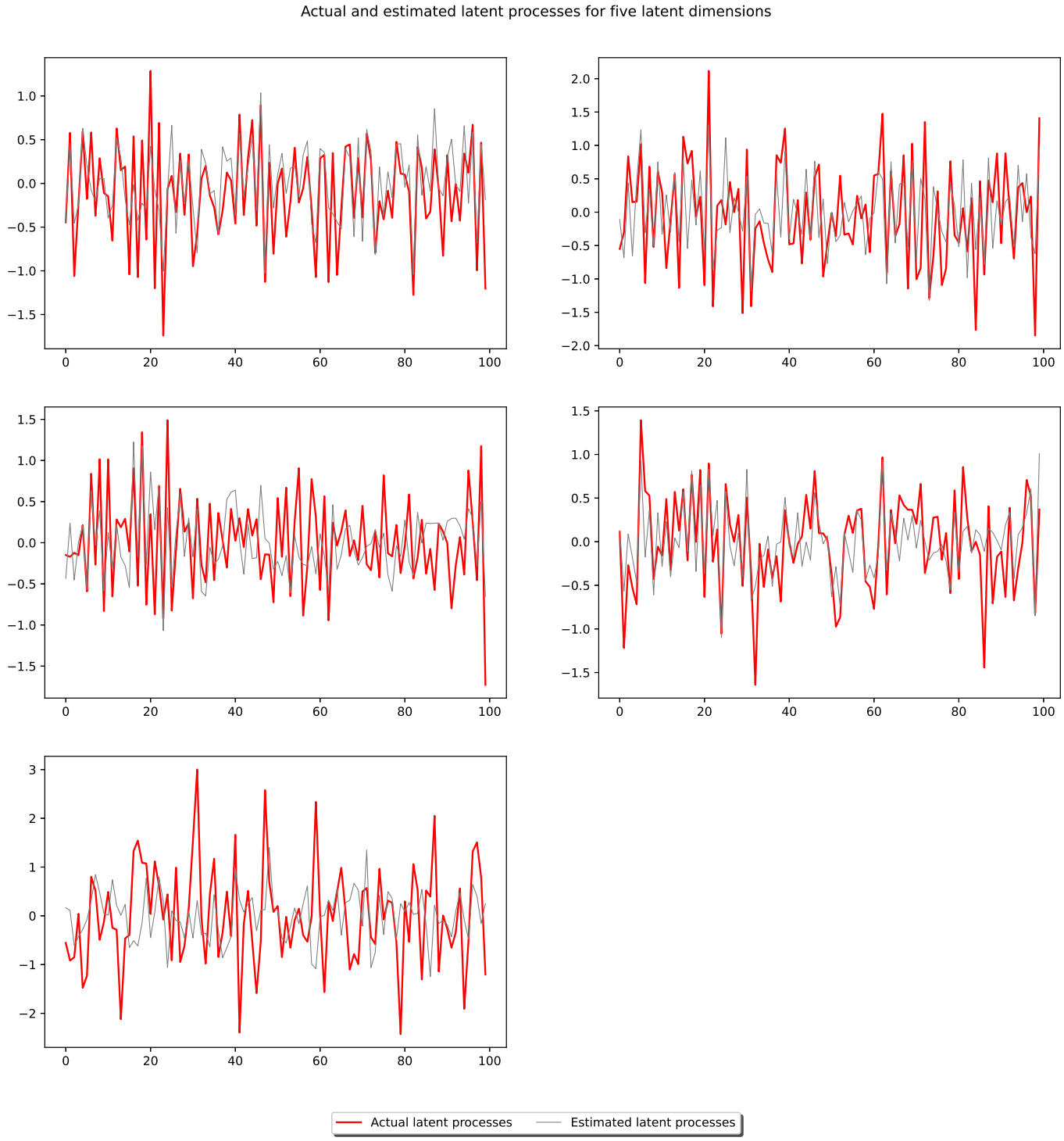


Fig. 11. Estimated latent processes $\widehat{\mathbf{x}}_t$ and actual latent processes $\mathbf{x}_t$ for all five dimensions.

E. Testing on real world data sets

We also assessed the performance of our model on five real-world data sets [46] and compared it to six state-of-the-art models. The suite of reference methods is composed of (1) two one-step ahead autoregressive GP models: GP-NARX [13] and NIGP [47], (2) three multi-step-ahead autoregressive and recurrent GP models in latent spaces: REVARB with one and two hidden layers [48] and MSGP [49], and (3) two GP-SSMs, based on a full Markovian state: SS-GP-SSM [50] and PR-SSM [46]. Specifically, the REVARB is a deep state-space model based on RNNs. Note that these benchmark methods are learned in an offline mode or in a batch mode, which means that they are trained on sets multiple times and then applied to test sets.

TABLE I
RMSEs AND STANDARD DEVIATIONS

	ONE-STEP-AHEAD, AUTOREGRESSIVE		MULTI-STEP-AHEAD, LATENT SPACE AUTOREGRESSIVE			MARKOVIAN STATE-SPACE MODELS		
TASK	GP-NARX	NIGP	REVARB 1	REVARB 2	MSGP	SS-GP-SSM	PR-SSM	RF-SSM
ACTUATOR	0.627	0.599	0.438	0.613	0.771	0.696	0.502	**0.295**
	(0.005)	(0)	(0.049)	(0.190)	(0.098)	(0.034)	(0.031)	**(0.037)**
BALLBEAM	0.284	0.087	0.139	0.209	0.124	411.6	**0.073**	0.107
	(0.222)	(0)	(0.007)	(0.012)	(0.034)	(273.0)	**(0.007)**	(0.010)
DRIVE	0.701	**0.373**	0.828	0.868	0.451	0.718	0.492	0.417
	(0.015)	**(0)**	(0.025)	(0.113)	(0.021)	(0.009)	(0.038)	(0.030)
FURNACE	1.201	1.205	1.195	1.188	1.277	1.318	1.249	**0.410**
	(0.000)	(0)	(0.002)	(0.001)	(0.127)	(0.027)	(0.029)	**(0.032)**
DRYER	0.310	0.268	0.851	0.355	**0.146**	**0.152**	**0.140**	0.273
	(0.044)	(0)	(0.011)	(0.027)	**(0.004)**	**(0.006)**	**(0.018)**	(0.021)

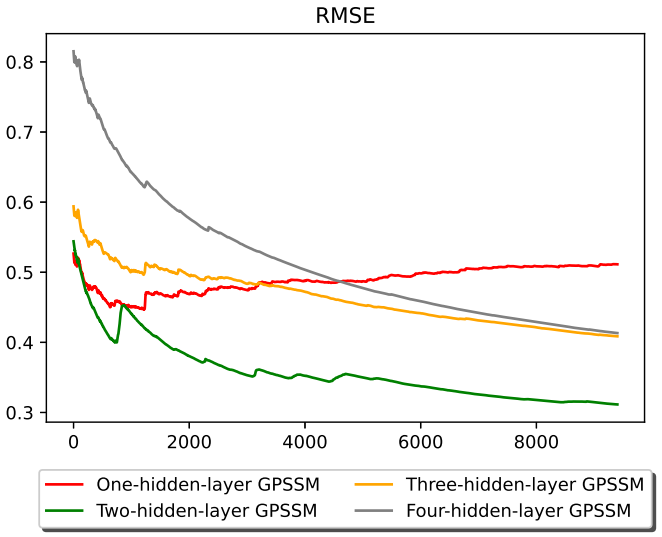


Fig. 12. RMSEs obtained from a single-hidden-layer GP-SSM to a four-hidden-layer GP-DSSM.

The settings of all the methods were the same, where the number of inducing points P or the number of random features J was 20. The latent dimension d_x was set to four. All the observations were normalized. The results of the best performer in terms of the Welch t-test are presented in bold numbers. Further information about our model including run time is provided in Table II. The training times were collected from a server with 12G RAM. The training time of our ensemble model depends on the number of candidate models, which was $S = 100$ in our experiments, and these models were trained separately. If they are deployed in a distributed manner, the training time in Table II can be reduced by around 100 times. The benchmark methods, however, can only be implemented by multiple iterations and cannot be conducted in a distributed way. The training and test times of the other methods are not provided in the corresponding papers and are affected by the used number of iterations in the computations.

As benchmarks, we have chosen to use batch methods due to lack of other sequential methods that operate under the assumption of unknown transition and observation functions, as described in our paper. In order to ensure a fair comparison, we utilized a test set with a small number of samples, specifically ranging from 100 to 500 samples. Opting for a smaller test set allows for a more gradual and less rapid change in the sequential method. The batch methods undergo multiple training iterations on the training set to ensure convergence of their parameters. By contrast, our method only exploited the training set once and thus, may have not achieved complete convergence by the end of the training process due to the small number of training samples. Even so, our method, RF-SSM, achieved the best performance on two data sets (*Actuator* and *Furnace*). It also was the second best performer on the data set *Drive* and the third best performer on the data set *Ballbeam*. All the results are shown in Table I. The numbers in parentheses are the standard deviations of the RMSEs among five runs with different seeds.

TABLE II
DATA INFORMATION

Task	N_{train}	N_{test}	$T_{train}(seconds)$	$T_{test}(seconds)$
ACTUATOR	512	512	701.0 (3.847)	13.8 (0.748)
BALLBEAM	500	500	689.6 (15.318)	20.6 (1.744)
DRIVE	250	250	307.6 (3.2)	4.2 (0.400)
FURNACE	148	148	160.4 (0.490)	3.8 (0.400)
DRYER	500	500	686.0 (6.841)	8.0 (0.00)

VII. CONCLUSIONS

In this paper, we addressed the problem of sequential estimation of state-space models and deep state-space models using Gaussian process-based state-space modeling and Gaussian process-based deep state-space modeling. An important advantage of the considered methodology is that it relaxes the assumption of knowing the functions in the observation and state equations. We implemented the Gaussian processes by using random feature-based Gaussian processes. The inference method is based on the combination of particle filtering and Bayesian linear regression. We also proposed an ensemble of filters for tracking the latent processes. With several experiments, we demonstrated the performance of the proposed method in different settings including synthetic examples and five real data sets. Further, we compared the performance of our method with 6 other state-of-the art

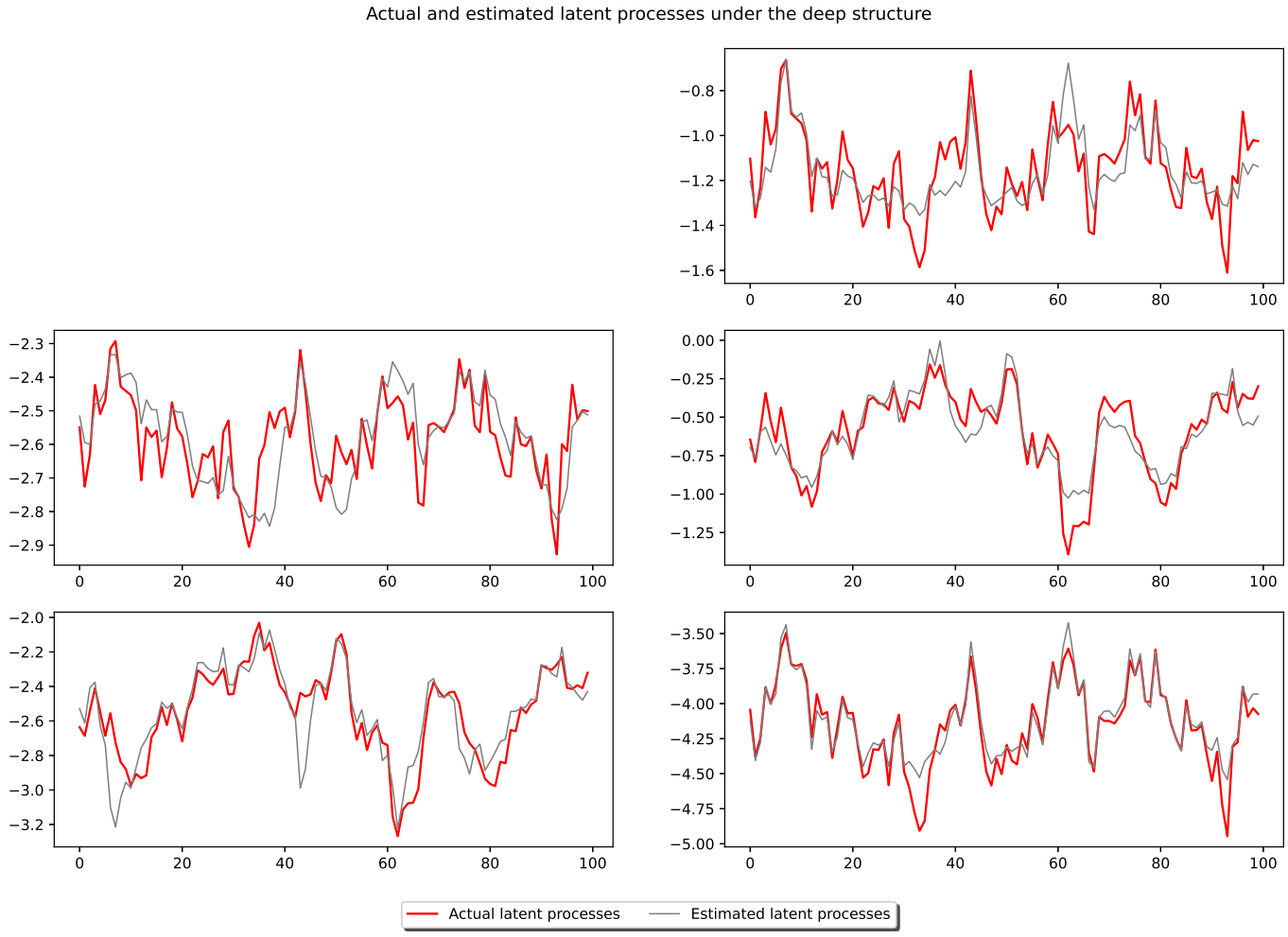


Fig. 13. On the left are the true values and estimates of $\mathbf{x}_{1,t}$, and on the right, the true values and estimates of $\mathbf{x}_{2,t}$.

methods. A limitation of our approach is that for too deep models, the method requires many more particles and a larger number of features. In future work, we will address the use of variational Bayes approach to acquire better set of random feature candidates so that we can reduce the number of features and number of particles. We will also consider applying more efficient sampling approaches.

REFERENCES

[1] J. Dean, D. Patterson, and C. Young, "A new golden age in computer architecture: Empowering the machine-learning revolution," IEEE Micro, vol. 38, no. 2, pp. 21–29, 2018.

[2] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, Deep Learning. MIT Press Cambridge, 2016.

[3] I. Katircioglu, B. Tekin, M. Salzmann, V. Lepetit, and P. Fua, "Learning latent representations of 3D human pose with deep neural networks," International Journal of Computer Vision, vol. 126, pp. 1–16, 2018.

[4] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," Neurocomputing, vol. 234, pp. 11–26, 2017.

[5] M. Vatsa, R. Singh, and A. Majumdar, Deep Learning in Biometrics. CRC Press, 2018.

[6] C. E. Rasmussen and C. K. Williams, Gaussian Processes for Machine Learning. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[7] M. Seeger, "Gaussian processes for machine learning," International Journal of Neural Systems, vol. 14, no. 02, pp. 69–106, 2004.

[8] J. Quiñonero Candela and C. E. Rasmussen, "A unifying view of sparse approximate Gaussian process regression," The Journal of Machine Learning Research, vol. 6, pp. 1939–1959, 2005.

[9] E. Snelson and Z. Ghahramani, "Local and global sparse Gaussian process approximations," in Artificial Intelligence and Statistics. PMLR, 2007, pp. 524–531.

[10] M. Titsias, "Variational learning of inducing variables in sparse Gaussian processes," in Artificial Intelligence and Statistics. PMLR, 2009, pp. 567–574.

[11] T. Beckers and S. Hirche, "Prediction with approximated Gaussian process dynamical models," IEEE Transactions on Automatic Control, 2021.

[12] R. Frigola, Y. Chen, and C. E. Rasmussen, "Variational Gaussian process state-space models," in Advances in Neural Information Processing Systems, 2014, pp. 3680–3688.

[13] J. Kocijan, A. Girard, B. Banko, and R. Murray-Smith, "Dynamic systems identification with Gaussian processes," Mathematical and Computer Modelling of Dynamical Systems, vol. 11, no. 4, pp. 411–424, 2005.

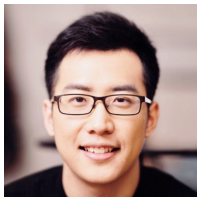
[14] J. Ko and D. Fox, "GP-Bayes filters: Bayesian filtering using Gaussian process prediction and observation models," Autonomous Robots, vol. 27, pp. 75–90, 2009.

[15] J. Ko, D. J. Klein, D. Fox, and D. Haehnel, "GP-UKF: Unscented Kalman filters with Gaussian process prediction and observation models," in 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2007, pp. 1901–1907.

[16] A. Svensson, A. Solin, S. Särkkä, and T. Schön, "Computationally efficient Bayesian learning of Gaussian process state space models," in Artificial Intelligence and Statistics. PMLR, 2016, pp. 213–221.

[17] J. Wang, A. Hertzmann, and D. J. Fleet, "Gaussian process dynamical

models," Advances in Neural Information Processing Systems, vol. 18, 2005.
[18] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 2, pp. 283–298, 2007.
[19] S. Eleftheriadis, T. Nicholson, M. Deisenroth, and J. Hensman, "Identification of gaussian process state space models," Advances in Neural Information Processing Systems, vol. 30, 2017.
[20] F. Tobar, P. M. Djurić, and D. P. Mandic, "Unsupervised state-space modeling using reproducing kernels," IEEE Transactions on Signal Processing, vol. 63, no. 19, pp. 5210–5221, 2015.
[21] D. Gedon, N. Wahlström, T. B. Schön, and L. Ljung, "Deep state space models for nonlinear system identification," IFAC-PapersOnLine, vol. 54, no. 7, pp. 481–486, 2021.
[22] A. Doucet, N. De Freitas, K. Murphy, and S. Russell, "Rao-Blackwellised particle filtering for dynamic Bayesian networks," arXiv preprint arXiv:1301.3853, 2013.
[23] C. Andrieu and A. Doucet, "Particle filtering for partially observed Gaussian state space models," Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 64, no. 4, pp. 827–836, 2002.
[24] R. Chen and J. S. Liu, "Mixture Kalman filters," Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 62, no. 3, pp. 493–508, 2000.
[25] M. Fraccaro, "Deep latent variable models for sequential data," Ph.D. dissertation, PhD thesis, Technical University of Denmark, 2018.
[26] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, "Deep state space models for time series forecasting," Advances in Neural Information Processing Systems, vol. 31, pp. 7785–7794, 2018.
[27] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," International Journal of Forecasting, vol. 36, no. 3, pp. 1181–1191, 2020.
[28] Y. Wang, A. Smola, D. Maddix, J. Gasthaus, D. Foster, and T. Januschowski, "Deep factors for forecasting," in International Conference on Machine Learning. PMLR, 2019, pp. 6607–6617.
[29] Z. Zhao, M. Emzir, and S. Särkkä, "Deep state-space Gaussian processes," arXiv preprint arXiv:2008.04733, 2020.
[30] M. Deisenroth and J. W. Ng, "Distributed Gaussian processes," in International Conference on Machine Learning. PMLR, 2015, pp. 1481–1490.
[31] E. Meeds and S. Osindero, "An alternative infinite mixture of Gaussian process experts," Advances in Neural Information Processing Systems, vol. 18, p. 883, 2006.
[32] C. E. Rasmussen and Z. Ghahramani, "Infinite mixtures of Gaussian process experts," Advances in Neural Information Processing Systems, vol. 2, pp. 881–888, 2002.
[33] V. Tresp, "Mixtures of Gaussian processes," Advances in Neural Information Processing Systems, pp. 654–660, 2001.
[34] C. Yuan and C. Neubauer, "Variational mixture of Gaussian process experts," in Advances in Neural Information Processing Systems, 2009, pp. 1897–1904.
[35] Q. Lu, G. Karanikolas, Y. Shen, and G. B. Giannakis, "Ensemble Gaussian processes with spectral features for online interactive learning with scalability," in International Conference on Artificial Intelligence and Statistics. PMLR, 2020, pp. 1910–1920.
[36] M. Lázaro-Gredilla, J. Quinonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal, "Sparse spectrum Gaussian process regression," The Journal of Machine Learning Research, vol. 11, pp. 1865–1881, 2010.
[37] A. Zellner, An Introduction to Bayesian Inference in Econometrics. John Wiley & Sons, 1971.
[38] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," IEEE Transactions on signal processing, vol. 50, no. 2, pp. 174–188, 2002.
[39] P. M. Djurić, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Míguez, "Particle filtering," IEEE Signal Processing Magazine, vol. 20, no. 5, pp. 19–38, 2003.
[40] A. Doucet and A. M. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," Handbook of Nonlinear Filtering, vol. 12, no. 656-704, p. 3, 2009.
[41] T. Li, M. Bolic, and P. M. Djurić, "Resampling methods for particle filtering: classification, implementation, and strategies," IEEE Signal Processing Magazine, vol. 32, no. 3, pp. 70–86, 2015.
[42] G. Gundersen, M. Zhang, and B. Engelhardt, "Latent variable modeling with random features," in International Conference on Artificial Intelligence and Statistics. PMLR, 2021, pp. 1333–1341.
[43] Y. Liu, M. Ajirak, and P. M. Djurić, "Inference with deep Gaussian process state space models," in 2022 30th European Signal Processing Conference (EUSIPCO). IEEE, 2022, pp. 792–796.
[44] K. Cutajar, E. V. Bonilla, P. Michiardi, and M. Filippone, "Random feature expansions for deep Gaussian processes," in International Conference on Machine Learning. PMLR, 2017, pp. 884–893.
[45] M. M. Dunlop, M. A. Girolami, A. M. Stuart, and A. L. Teckentrup, "How deep are deep Gaussian processes?" Journal of Machine Learning Research, vol. 19, no. 54, pp. 1–46, 2018.
[46] A. Doerr, C. Daniel, M. Schiegg, N.-T. Duy, S. Schaal, M. Toussaint, and T. Sebastian, "Probabilistic recurrent state-space models," in International conference on machine learning. PMLR, 2018, pp. 1280–1289.
[47] A. McHutchon and C. Rasmussen, "Gaussian process training with input noise," Advances in Neural Information Processing Systems, vol. 24, 2011.
[48] C. L. C. Mattos, Z. Dai, A. Damianou, G. A. Barreto, and N. D. Lawrence, "Deep recurrent Gaussian processes for outlier-robust system identification," Journal of Process Control, vol. 60, pp. 82–94, 2017.
[49] A. Doerr, C. Daniel, D. Nguyen-Tuong, A. Marco, S. Schaal, T. Marc, and S. Trimpe, "Optimizing long-term predictions for model-based policy search," in Conference on Robot Learning. PMLR, 2017, pp. 227–238.
[50] A. Svensson and T. B. Schön, "A flexible state–space model for learning nonlinear dynamical systems," Automatica, vol. 80, pp. 189–199, 2017.



Yuhao Liu received the B.Sc. degree in mathematical sciences from Tsinghua University, Beijing, China, in 2016, the M.Sc. degree in statistics from the University of Michigan, Ann Arbor, MI, USA, in 2018, and the Ph.D. degree in applied mathematics and statistics from Stony Brook University, NY, USA, in 2023. He is currently working as a data scientist with Capital One, McLean, VA, USA. His research interests include Bayesian deep learning, ensemble machine learning, signal processing, causal inference, and their applications in finance and neuroscience.

Marzieh Ajirak is a Ph.D. candidate in the Department of Electrical and Computer Engineering at Stony Brook University, NY, USA. She holds both B.Sc. and M.Sc. degrees in Electrical Engineering from Isfahan University of Technology, Iran. Marzieh's research expertise lies in the theory of machine learning, with a particular focus on its applications in various biomedical challenges.

Petar M. Djurić received the B.S. and M.S. degrees in electrical engineering from the University of Belgrade, Belgrade, Serbia, and the Ph.D. degree in electrical engineering from the University of Rhode Island, South Kingstown, RI, USA. He is currently a SUNY Distinguished Professor and the Chair of the Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY, USA. From 2008 to 2009, he was a Distinguished Lecturer of the IEEE Signal Processing Society. His research interests include signal and information processing and machine learning. Prof. Djurić was the recipient of the EURASIP Technical Achievement Award in 2012. In 2008, he was the Chair of Excellence of Universidad Carlos III de Madrid-Banco de Santander. He served on numerous committees of the IEEE Signal Processing Society and many professional conferences and workshops. He was the founding Editor-in-Chief of the IEEE TRANSACTIONS ON SIGNAL AND INFORMATION PROCESSING OVER NETWORKS. In 2022, he was elected as a foreign Member of the Serbian Academy of Engineering Sciences. Prof. Djurić is also a Fellow of EURASIP.