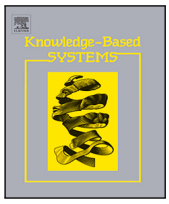
Knowledge-Based Systems 314 (2025) 113187

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



Fully Bayesian differential Gaussian processes through stochastic differential equations☆

Jian Xu [a], Zhiqi Lin [a], Min Chen [a], Junmei Yang [a], Delu Zeng [a,*], John Paisley [b]

[a] South China University of Technology, Guangzhou, China
[b] Columbia University, NY, USA

ARTICLE INFO

Keywords:
Differential Gaussian process
Variational inference
Stochastic differential equations

ABSTRACT

Deep Gaussian process models typically employ discrete hierarchies, but recent advancements in differential Gaussian processes (DiffGPs) have extended these models to infinite depths. However, existing DiffGP approaches often overlook the uncertainty in kernel hyperparameters by treating them as fixed and time-invariant, which degrades the model's predictive performance and neglects the posterior distribution. In this work, we introduce a fully Bayesian framework that models kernel hyperparameters as random variables and utilizes coupled stochastic differential equations (SDEs) to jointly learn their posterior distributions alongside those of inducing points. By incorporating the estimation uncertainty of hyperparameters, our method significantly enhances model flexibility and adaptability to complex dynamic systems. Furthermore, we employ a black-box adaptive SDE solver with a neural network to achieve realistic, time-varying posterior approximations, thereby improving the expressiveness of the variational posterior. Comprehensive experimental evaluations demonstrate that our approach outperforms traditional methods in terms of flexibility, accuracy, and other key performance metrics. This work not only provides a robust Bayesian extension to DiffGP models but also validates its effectiveness in handling intricate dynamic behaviors, thereby advancing the applicability of Gaussian process models in diverse real-world scenarios.

1. Introduction

Gaussian Process (GP) models [1] are non-parametric Bayesian methods widely used for tasks such as regression, classification, and optimization. They provide a flexible way to model uncertainty by assuming a joint distribution over the input data and capturing correlations between data points through a kernel function. GPs have found extensive applications in forecasting, including predicting agricultural commodity prices [2,3], surrogate modeling [4], Bayesian optimization [5], and reinforcement learning [6]. However, despite their versatility, GP models face challenges when dealing with non-Gaussian distributions, complex distributions, time series data, and other difficult tasks.

To address this, deep Gaussian processes (DGPs) [7] have been introduced for more expressive representations. However, DGPs may also face learning issues if the individual GPs are not invertible [8,9]. One approach to address this challenges is via differential GPs (DiffGPs) [10], which model data evolution in continuous time using systems of stochastic differential equations. This approach enables the learning of continuous-time transformations of the data, providing a more intuitive means of capturing data dynamics compared to conventional techniques. DiffGPs warp inputs through differential fields to generalize discrete layers into a dynamic system. The intuition of "warping" the inputs over time, as derived from the original DiffGP paper, is an extension of the concept of deep Gaussian processes in continuous layers. Similar to the transition from ResNet [11] to neural ODEs [12], this method seeks to adaptively transform the input space over time, enabling the model to more effectively capture intricate patterns and dependencies within the data.

DiffGP methods [10] often overlook the uncertainty in kernel hyperparameters, which is crucial for accurate model fitting and reliable uncertainty estimation. This uncertainty refers to the process of estimating and selecting these hyperparameters, which directly impacts the model's predictive performance and the posterior distribution. The choice of covariance function in Gaussian processes significantly influences the shape of the posterior distribution and the uncertainty of predictions, highlighting the importance of selecting an appropriate

☆ The work is supported by the Fundamental Research Program of Guangdong, China, under Grant 2023A1515011281.
* Corresponding author.
E-mail addresses: 202010106028@mail.scut.edu.cn (J. Xu), 202311089192@mail.scut.edu.cn (Z. Lin), minchen@ieee.org (M. Chen), yjunmei@scut.edu.cn (J. Yang), dlzeng@scut.edu.cn (D. Zeng), jwp2128@columbia.edu (J. Paisley).

https://doi.org/10.1016/j.knosys.2025.113187
Received 16 November 2024; Received in revised form 7 February 2025; Accepted 14 February 2025
Available online 25 February 2025

covariance function [13,14].

In Gaussian process modeling, model selection involves not only choosing the covariance function but also estimating the hyperparameters. Traditionally, this process is performed by maximizing the marginal likelihood or its lower bound. However, the marginal likelihood is typically non-convex, especially when there are multiple local optima or when hyperparameters are weakly identifiable. This non-convexity makes optimization challenging, as gradient-based methods can get stuck in local minima, leading to overfitting and underestimating prediction uncertainty. Moreover, the sensitivity of gradient-based optimization to initial values further complicates the maximization of the marginal likelihood.

To overcome these issues, improved methods are needed to enhance the accuracy of hyperparameter estimation, which in turn will lead to better modeling of uncertainty and prevent overfitting. In this work, we propose a fully Bayesian approach to DiffGPs, treating kernel hyperparameters as random variables and using coupled stochastic differential equations (SDEs) to learn their posterior distribution and that of inducing points. This method effectively addresses the non-convexity of the marginal likelihood, providing robust parameter estimation. By incorporating uncertainty in hyperparameters through the posterior distribution, overfitting is avoided and generalization to new data improved. Bayesian methods offer a comprehensive understanding of parameter estimation by capturing inherent uncertainty, improving model reliability and adaptability.

Extending the Bayesian approach to include hyperparameters within a hierarchical framework increases the complexity of posterior computations, making the process more challenging. We introduce a novel methodology using SDEs to learn the posterior distribution of hyperparameters and inducing points, capturing their uncertainty effectively. By integrating Bayesian inference with SDEs, our approach enhances model adaptability and robustness in capturing system dynamics.

Our work introduces a novel network architecture that not only enhances the expressiveness of DiffGPs, but also integrates the uncertainty of kernel parameters and inducing point distributions into a fully Bayesian inference framework, incorporating their time-correlated posterior SDEs. By leveraging state-of-the-art SDE gradient estimators, such as those in [15–18], we demonstrate the effectiveness of approximate inference by maximizing our modified variational lower bound, significantly improving the scalability of gradient-based variational inference compared to previous studies. Computation of the output layer state is simplified using a black-box adaptive SDE solver, simplifying the modeling process and enhancing the efficiency of our proposed methodology.

Our approach offers two distinct advantages over previous DiffGP models: (1) It incorporates the uncertainty of inducing points and kernel hyperparameters within a fully Bayesian framework, which enhances flexibility and adaptability in capturing complex dynamics. (2) The SDE solver is responsible for describing how the posterior distributions of the kernel hyperparameters and inducing points evolve over time. By adopting an adaptive SDE approach, we achieve a comprehensive and realistic approximation of the posterior, ensuring that this approximation can effectively capture the dynamics of complex systems. This results in a more accurate posterior approximation that helps prevent overfitting.

Experimental evaluation demonstrates the improvement of our proposed method over traditional approaches in terms of flexibility, accuracy, and other metrics. Our contributions are outlined as follows:

- We introduce a fully Bayesian approach that treats kernel hyperparameters as random variables and utilizes coupled stochastic differential equations (SDEs) to learn their posterior distribution and that of inducing points for DiffGPs.
- By using an adaptive SDE method with a neural network as a black-box solver, we achieve a realistic posterior approximation that effectively captures time-varying dynamics and enhances the expressiveness of the variational posterior.
- Experimental results show the advantage of our method over traditional approaches in terms of flexibility, accuracy, and other metrics.

The structure of this paper is organized as follows. Section 2 introduces the foundational models, including Gaussian Processes, Sparse Representations, and Continuous-time Gaussian Processes (DiffGPs). Section 3 details our proposed method, called Fully Bayesian Differential Gaussian Processes (FB-Diff), along with the underlying methodologies and implementation details. Section 4 reviews related work and provides a comprehensive literature survey. Section 5 presents the experimental setup and results, demonstrating the effectiveness of our approach. Finally, Section 6 concludes the paper and discusses potential directions for future research.

2. Model

2.1. Gaussian processes and sparse representation

Gaussian processes (GPs) are powerful probabilistic models that define a distribution over functions. Given a set of input points $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]$, where each $x_i \in \mathbb{R}^D$, a GP specifies a joint Gaussian distribution over the corresponding function values $\mathbf{f} = [f(\mathbf{x}_1), \dots, f(\mathbf{x}_N)]^\top \in \mathbb{R}^N$. The fundamental property of GPs is that the outputs at different points correlate based on their similarity, as measured by a kernel function $k(\mathbf{x}, \mathbf{x}')$ [1].

Traditionally, a zero-mean Gaussian process prior is defined on a function $f(\mathbf{x})$ over vector inputs $\mathbf{x} \in \mathbb{R}^D$:

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}')), \tag{1}$$

where $k(\mathbf{x}, \mathbf{x}')$ is the kernel function that captures the covariance between function values at different inputs.

To handle large datasets effectively, sparse Gaussian processes employ a small set of inducing or landmark variables [19–22]. These inducing variables $\mathbf{u} = u_1, \dots, u_M{}^\top \in \mathbb{R}^M$, where M is much smaller than N, can be selected from the dataset or chosen independently. For example, Tran et al. (2021) [23] propose utilizing neighbor information to guide the selection of inducing points, while Jafrasteh et al. (2022) [24] suggest an input-dependent approach where inducing points are determined based on the input data characteristics. By conditioning the GP prior on these inducing variables $\mathbf{u}$ and their corresponding locations $\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_M\}^\top$, we can obtain posterior predictions at the data points.

The sparse GP posterior predictions, given the inducing variables $\mathbf{u}$ and their locations $\mathbf{Z}$, are given by

$$\begin{aligned} \mathbf{f} \mid \mathbf{u}, \mathbf{Z} &\sim \mathcal{N}(\mathbf{Qu}, \mathbf{K_{XX}} - \mathbf{QK_{ZZ}Q}^\top) \\ \mathbf{u} &\sim \mathcal{N}(\mathbf{0}, \mathbf{K_{ZZ}}). \end{aligned} \tag{2}$$

Here, $\mathbf{Q} = \mathbf{K_{XZ}}(\mathbf{K_{ZZ}} + \sigma_n^2\mathbf{I})^{-1}$ represents the matrix of coefficients linking the inducing variables to the function values. In this expression, the subscript n denotes the index of the sample point, $\mathbf{K_{ZZ}}$ is the pairwise kernel matrix for the inducing points, $\mathbf{K_{XX}}$ is the kernel matrix for the data points, $\mathbf{K_{XZ}}$ is the kernel matrix between the data points and the inducing points, and σ_n^2 is the noise variance of the observations.

The main inference problems for Gaussian processes are related to the inducing points $\mathbf{Z}$, inducing variables $\mathbf{u}$, and the kernel hyperparameters λ found in, for example, the classic Gaussian kernel,

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2l^2}\right). \tag{3}$$

For model learning, variational inference (VI) [25] and Markov Chain Monte Carlo (MCMC) [26] have been widely used. By incorporating sparse representation through inducing variables, we can effectively model and make predictions in large-scale datasets using Gaussian processes.

2.2. Continuous-time Gaussian processes

The continuous-time deep learning paradigm introduced by [10] focuses on a model called DiffGP, which is a continuous-time deep Gaussian process model through infinite, infinitesimal differential compositions. In DiffGP models, the input data is transformed using a stochastic differential equation (SDE) flow to obtain transformed inputs $\mathbf{X}_T$, which are then used for model fitting after a predefined time T. The parameter T governs the flow's length and the system's capacity, akin to the number of layers in traditional deep neural networks or deep GPs.

In this framework, the inputs are redefined as temporal functions $\mathbf{x} : \mathbb{T} \to \mathbb{R}^D$ over time, with state paths $\mathbf{x}_t$ evolving over $t \in \mathbb{T} = \mathbb{R}_+$. The observed inputs $\mathbf{x}_{i,0}$ represent the initial states. The goal is to classify or regress the final data points $\mathbf{X}_T = \left(\mathbf{x}_{1,T}, \ldots, \mathbf{x}_{N,T}\right)^\top$, which represent the states after T time steps of an SDE flow, using a Gaussian process predictor. In the above notation, the first index represents the dimension of the sample points, indicating the number of sample particles. The second index represents the dimension of the discrete time points. The predictor, denoted as $g\left(\mathbf{x}_T\right)$, is assumed to follow a Gaussian process prior with zero mean and covariance function $K\left(\mathbf{x}_T, \mathbf{x}'_T\right)$. When $T = 0$, the framework simplifies to a standard Gaussian process.

The prediction relies on the structure of the final dataset $\mathbf{X}_T$, which is determined by the SDE flow $d\mathbf{x}_t$ originating from the initial data $\mathbf{X}$. We focus on SDE flows of the Ito type,

$$d\mathbf{x}_t = \boldsymbol{\mu}\left(\mathbf{x}_t\right) dt + \sqrt{\boldsymbol{\Sigma}\left(\mathbf{x}_t\right)} dW_t, \tag{4}$$

where

$$\boldsymbol{\mu}(\mathbf{x}_t) = \mathbf{K}_{\mathbf{xZ}}\mathbf{K}_{\mathbf{ZZ}}^{-1}\,\mathrm{vec}\,(\mathbf{U}),$$
$$\boldsymbol{\Sigma}(\mathbf{x}_t) = \mathbf{K}_{\mathbf{xx}} - \mathbf{K}_{\mathbf{xZ}}\mathbf{K}_{\mathbf{ZZ}}^{-1}\mathbf{K}_{\mathbf{Zx}}.$$

The vector-valued Gaussian process is conditioned on the inducing variables $\mathbf{U} = \left(\mathbf{u}_1, \ldots, \mathbf{u}_M\right)^\top$, which define the function values $\mathbf{f}(\mathbf{z})$ at the inducing points $\mathbf{Z} = \left(\mathbf{z}_1, \ldots, \mathbf{z}_M\right)$. In Eq. (4), $\mathbf{U}$ has dimensions $M \times D$, corresponding to a multi-output Gaussian process with output dimension D. As a result, the dimension of $\mathbf{K}_{\mathbf{ZZ}}$ is $MD \times MD$, representing a block matrix of kernels that capture the dependencies between the M inducing points and the D output dimensions.

The transformation process can be simulated using an SDE solver, such as Euler discretization [27],

$$\mathbf{x}_T = \mathbf{x}_0 + \int_0^T \boldsymbol{\mu}(\mathbf{x}_t)\, dt + \int_0^T \sqrt{\Sigma\left(\mathbf{x}_t\right)}\, dW_t. \tag{5}$$

The DiffGP model, which captures the time evolution and dynamics of data in a continuous manner, offers a more natural and flexible approach compared to traditional discretized approaches like Deep GPs [7,28]. However, the original DiffGP model has limitations. For example, it treats the kernel parameters as point estimates and does not consider their uncertainty using Bayesian inference methods. Additionally, it does not account for the temporal variability of the kernel parameters and inducing points, similar to how each layer in a deep GP has its own kernel parameters and inducing points. In the following section, we will introduce a new fully Bayesian framework that addresses these issues by utilizing recent advancements in SDE theory and its connection to variational inference.

3. Fully Bayesian variational inference

3.1. Modeling uncertainty in kernel hyperparameters and inducing points

For Gaussian process (GP) regression models, the effectiveness of the model is closely tied to the selected kernel hyperparameters and inducing points, an inherently complex task. Finding optimal values for these parameters presents a challenge. Our proposed methodology employs a Bayesian framework to treat the uncertainty inherent in choosing kernel hyperparameters and inducing points.

In the generative process of the model, we assume prior distributions on the kernel hyperparameters λ_t, inducing points $\mathbf{Z}_t$, and inducing variables $\mathbf{U}_t$. These various parameters control the behavior of the Gaussian process model and are crucial for determining the smoothness of the model. The inducing points $\mathbf{Z}_t$ are any set of points within the input space that can be optimized to approximate the function values at all possible input locations. The inducing variables $\mathbf{U}_t$ are the corresponding function values learned for the inducing points.

Given these prior distributions, the model generates the observed data $\mathbf{X}$ and $\mathbf{y}$ by drawing samples from the Gaussian process defined by the kernel function with the hyperparameters λ_t and the inducing variables $\mathbf{U}_t$. For example, in the classic Gaussian kernel, $\lambda = \{\sigma_f, l\}$. The posterior distribution of the hyperparameters, inducing points, and inducing variables can then be estimated based on the observed data using Bayesian inference techniques. This allows us to make predictions and infer the underlying structure of the data based on the Gaussian process model,

$$\begin{aligned}
\text{Prior over hyperparameters}: &\quad \lambda_t \sim p(\lambda_t)\\
\text{Prior over inducing points}: &\quad \mathbf{Z}_t \sim p(\mathbf{Z}_t)\\
\text{Prior over inducing variables}: &\quad \mathbf{U}_t \sim \mathcal{GP}(0, K(\mathbf{Z}_t, \mathbf{Z}'_t))\\
\text{Model forward SDE}: &\quad d\mathbf{x}_t = \boldsymbol{\mu}\left(\mathbf{x}_t\right) dt + \sqrt{\Sigma\left(\mathbf{x}_t\right)} dW_t\\
\text{Predictor Gaussian process}: &\quad \mathbf{g} \sim \mathcal{GP}(0, K\left(\mathbf{x}_T, \mathbf{x}'_T\right))\\
\text{Data likelihood}: &\quad \mathbf{y} \mid \mathbf{g} \sim \mathcal{N}\left(\mathbf{g}, \sigma_n^2 \mathbb{I}\right)
\end{aligned} \tag{6}$$

Inspired by the ANODEV2 method [29], which extends a complex ODE model for evolving neural network parameters, we view the hyperparameters λ_t and inducing points $\mathbf{Z}_t$ as prior processes that evolve over time. In contrast to traditional fully-GP models and previous works like DiffGPs, this approach departs from the fixed hyperparameters assumption and allows for a more dynamic modeling of the parameter evolution. Drawing parallels to evolutionary computing methods such as HyperNEAT [30], Compressed Weight Search [31], and Hypernetworks [32], which employ secondary networks to generate parameters for the main network, our approach extends this concept to parameter evolution in GP-SDE models. By adopting a Bayesian perspective, we aim to capture the time-varying distribution of hyperparameters, enhancing the adaptability of our model.

We can apply the concept of amortized variational inference, as introduced in the works of [13,33,34], to optimize the factorized approximate posterior distribution $q(\lambda_t, \mathbf{Z}_t, \mathbf{U}_t) = q(\lambda_t)q(\mathbf{Z}_t)q(\mathbf{U}_t)$. This requirement is met by employing the mean-field assumption, where the posterior is approximated as a product of several independent factors. The goal is to minimize the Kullback–Leibler (KL) divergence between this approximate posterior and the true posterior distribution. This optimization objective is equivalent to maximizing the Evidence Lower Bound (ELBO), which serves as a lower bound on the marginal likelihood of the data under the model. By maximizing the ELBO, we can efficiently approximate the true posterior distribution and make accurate inference about the model's parameters and latent variables,

$$\begin{aligned}
\log p(\mathbf{y}) \geq\ & \mathbb{E}_{q(\lambda)q(\mathbf{Z})q(\mathbf{U})p(\mathbf{x}|\lambda,\mathbf{Z},\mathbf{U})p(\mathbf{g}|\mathbf{x})}[\log p(\mathbf{y} \mid \mathbf{g})]\\
& - \mathrm{KL}\left(q\left(\lambda\right) \parallel p\left(\lambda\right)\right)\\
& - \mathrm{KL}\left(q\left(\mathbf{Z}\right) \parallel p\left(\mathbf{Z}\right)\right)\\
& - \mathrm{KL}\left(q\left(\mathbf{U}\right) \parallel p\left(\mathbf{U}\right)\right),
\end{aligned} \tag{7}$$

where λ, $\mathbf{Z}$, and $\mathbf{U}$ represent the trajectories of λ_t, $\mathbf{Z}_t$ and $\mathbf{U}_t$ for $t \in [0, T]$. We next demonstrate how to efficiently perform variational inference by optimizing the ELBO to estimate the prior hyperparameters and the parameters of a tractable approximate posterior.

3.2. Approximate posterior through latent stochastic differential equations

To perform posterior inference in our model, we leverage latent stochastic differential equations [15,35–37]. These SDEs provide a

principled and flexible framework for modeling complex temporal dynamics. Specifically, we can represent both the prior and the approximate posterior of λ_t and $\mathbf{Z}_t$ using coupled SDEs in the following system of differential equations,

$$\begin{cases} d\lambda_t = h_{\theta_\lambda}(\lambda_t, t)\,dt + l_\lambda(\lambda_t, t)\,dW_t & \text{(prior)} \\ d\lambda_t = h_{\phi_\lambda}(\lambda_t, t)\,dt + l_\lambda(\lambda_t, t)\,\mathrm{d}W_t & \text{(posterior approx)} \\ d\mathbf{Z}_t = h_{\theta_z}(\mathbf{Z}_t, t)\,dt + l_z(\mathbf{Z}_t, t)\,dW_t & \text{(prior)} \\ d\mathbf{Z}_t = h_{\phi_z}(\mathbf{Z}_t, t)\,dt + l_z(\mathbf{Z}_t, t)\,\mathrm{d}W_t & \text{(posterior approx)} \end{cases} \tag{8}$$

The difference between the prior and posterior processes lies solely in their drift terms. Similar to classical Bayesian analysis, the parameters of the prior SDE can be set to simple, fixed values, such as constants or basic affine transformations, while the drift term of the posterior SDE is parameterized by a fully connected neural network. This allows us to compute the KL divergence between the prior and posterior SDEs using Girsanov's theorem [38]. We apply this approach to both $\mathbf{Z}$ and λ. Although we currently assume the drift term to be a simple function, it is also feasible to employ complex networks to learn the prior.

In the context of an Ornstein–Uhlenbeck (OU) prior stochastic differential equation (SDE), we specify fixed prior drift coefficients as $h_{\theta_\lambda} = -\lambda_t$ and $h_{\theta_z} = -\mathbf{Z}_t$, along with fixed prior diffusion coefficients as $l_\lambda = \sigma_\lambda \mathbb{I}$ and $l_z = \sigma_z \mathbb{I}$. We choose the OU process because of its simplicity and well-studied properties. This process can be represented by the following SDE,

$$\begin{aligned} d\lambda_t &= -\lambda_t dt + \sigma_\lambda dW_t \quad \text{(prior)} \\ d\mathbf{Z}_t &= -\mathbf{Z}_t dt + \sigma_z dW_t \quad \text{(prior)} \end{aligned} \tag{9}$$

For the approximate posterior processes of λ_t and $\mathbf{Z}_t$, we also employ an SDE representation, where h_{ϕ_λ} and h_{ϕ_z} are parameterized neural networks, and all drift and diffusion functions are Lipschitz continuous. Due to these drifts, the approximate posterior process will typically exhibit non-Gaussian, non-factorized marginals. It may seem restrictive to assume that the diffusion terms of the prior and posterior are identical in Eq. (8). However, previous results from the Neural SDE literature demonstrate that any posterior can be approximated with arbitrary closeness using such a functional form given a sufficiently expressive drift process [15,18,37,39].

As a result, the Kullback–Leibler (KL) divergence between these distributions is finite and can be estimated by sampling paths from the posterior process [15,35]. Using Girsanov's theorem, which states how probability measures change in stochastic processes under a change of drift, we can express the ELBO in a concise form,

$$\begin{aligned} \log p(\mathbf{y}) \geq \ \mathrm{ELBO} &= \mathbb{E}_{q(\lambda)q(\mathbf{Z})q(\mathbf{U})p(\mathbf{x}|\lambda,\mathbf{Z},\mathbf{U})p(\mathbf{g}|\mathbf{x})}[\log p(\mathbf{y} \mid \mathbf{g})] \\ &\quad - \frac{1}{2}\int_0^T \mathbb{E}_{q(\lambda_t)}[\left|u_\lambda(\lambda_t, t)\right|^2]dt \\ &\quad - \frac{1}{2}\int_0^T \mathbb{E}_{q(\mathbf{Z}_t)}[\left|u_z(\mathbf{Z}_t, t)\right|^2]dt - \mathrm{KL}(q(\mathbf{U}) \parallel p(\mathbf{U})) \end{aligned} \tag{10}$$

where

$$u_\lambda(\lambda_t, t) = l_\lambda(\lambda_t, t)^{-1}\left(h_{\theta_\lambda}(\lambda_t, t) - h_{\phi_\lambda}(\lambda_t, t)\right),$$
$$u_z(\mathbf{Z}_t, t) = l_z(\mathbf{Z}_t, t)^{-1}\left(h_{\theta_z}(\mathbf{Z}_t, t) - h_{\phi_z}(\mathbf{Z}_t, t)\right).$$

The terms $l_\lambda(\lambda_t, t)^{-1}$ and $l_z(\mathbf{Z}_t, t)^{-1}$ represent the left inverses, with the expectation being taken over the approximate posterior process defined by Eq. (8). The functions u_λ and u_z need to fulfill the Novikov condition [15], which ensures that the drift and diffusion terms of the SDE are well-defined and that the process remains consistent with the requirements of stochastic calculus.

All estimates of stochastic differential equation (SDE) paths and gradients are simulated and computed using state-of-the-art SDE solvers, as outlined in the work by [15]. Furthermore, to address large-scale data challenges efficiently, we can utilize mini-batch surrogates for likelihood optimization. This approach uses ideas from backpropagation introduced in [40] and stochastic optimization techniques such as those discussed in [28,41,42]. By employing mini-batch surrogates, we can optimize likelihood functions for models handling significant amounts of data while retaining computational efficiency,

$$\log p(\mathbf{y}|\mathbf{g}) = \sum_{i=1}^{N} \log p(y_i|\mathbf{g}) \approx \frac{N}{B}\sum_{i=1}^{B} \log p(y_i|\mathbf{g}). \tag{11}$$

For variational inference of the inducing variable $\mathbf{U}_t$, we follow the approach of [10] by assuming that its posterior is a Gaussian distribution,

$$q(\mathbf{U}_t) = \mathcal{N}(\boldsymbol{m}_t, \boldsymbol{S}_t). \tag{12}$$

What differentiates this paper from prior research is that $\boldsymbol{m}_t$ and $\boldsymbol{S}_t$ are time series while in the original DiffGP, this term is modeled as a constant. We aim to characterize the dynamical system of the inducing variables over time. Because the Kullback–Leibler divergence between two Gaussian distributions is analytical, the expression $\mathrm{KL}(q(\mathbf{U}_t) \parallel p(\mathbf{U}_t))$ can be explicitly written as,

$$\begin{aligned} \mathrm{KL}(q(\mathbf{U}) \parallel p(\mathbf{U})) &= \int_0^T \mathbb{E}_{q(\mathbf{U}_t, \lambda_t, \mathbf{Z}_t)} \log \frac{q(\mathbf{U}_t)}{p(\mathbf{U}_t)} dt \\ &= \frac{1}{2}\int_0^T \mathbb{E}_q\left[\mathrm{trace}\left(K_{\mathbf{Z}_t\mathbf{Z}_t}^{-1}\boldsymbol{S}_t\right) + \boldsymbol{m}_t^T K_{\mathbf{Z}_t\mathbf{Z}_t}^{-1}\boldsymbol{m}_t + \ln\frac{|K_{\mathbf{Z}_t\mathbf{Z}_t}|}{|\boldsymbol{S}_t|}\right] dt \end{aligned} \tag{13}$$

In the second line, since $\mathbf{U}_t$ has already been integrated out, there is no need for an expectation with respect to $\mathbf{U}_t$. However, the expectations with respect to $\mathbf{Z}_t$ and λ_t are still necessary.

3.3. Simulation and predictions

By combining Eq. (4) with Eq. (8), we have

$$d\begin{pmatrix} \mathbf{x}_t \\ \lambda_t \\ \mathbf{Z}_t \end{pmatrix} = \begin{pmatrix} \boldsymbol{\mu}(\mathbf{x}_t) \\ h_{\phi_\lambda}(\lambda_t, t) \\ h_{\phi_z}(\mathbf{Z}_t, t) \end{pmatrix} dt + \begin{pmatrix} \sqrt{\Sigma(\mathbf{x}_t)} \\ l_\lambda(\lambda_t, t) \\ l_z(\mathbf{Z}_t, t) \end{pmatrix} dW_t. \tag{14}$$

By leveraging advanced SDE solvers for state trajectory approximation, we can perform stochastic gradient estimation to optimize Eq. (10). This technique enhances our understanding of the model's loss function and enables more efficient optimization. Integrating Bayesian methodologies for hyperparameters and inducing points with dynamic SDE posterior estimation leads to a more flexible and expressive posterior estimation for DiffGPs. This advancement improves the model's adaptability in capturing complex system dynamics and enhances predictive capabilities, providing a tool for robust model predictions. We present the algorithmic framework in Algorithm 1. As described in Section 2.1, the GP sparse representation method significantly reduces the model complexity by employing the sparse inducing points approach [19–22] in the DIffGP model. This decreases the computational complexity from $\mathcal{O}(M^3)$ to $\mathcal{O}(NM^2)$, where M is the number of inducing points $\mathbf{Z}$, which is much smaller than N.

In summary, our method leverages the simulation of state trajectories as outlined in Eq. (14), allowing us to sample the entire model's predictive values. The integration of posterior parameter estimation through the SDE method provides a significant advantage over traditional approaches, enhancing the model's robustness and improving uncertainty estimation. This approach improves the adaptability of the model to capture complex system dynamics as well as its predictive ability.

4. Related work

Gaussian Processes (GPs). GPs are a powerful class of non-parametric models widely used for regression, classification, and Bayesian optimization. They provide a probabilistic framework for modeling data by assuming that any finite subset of observations follows a multivariate Gaussian distribution. The flexibility of GPs allows them to capture complex patterns and uncertainty in data, making them ideal

Algorithm 1 FB-DiffGP Algorithm

Require: $\mathbf{X}$: input data matrix; $\mathbf{Y}$: labels; B: mini-batch size; E: number of epochs; η: learning rate; Parameter initialization for ϕ_λ, ϕ_z, $\boldsymbol{m}_t$, $\boldsymbol{S}_t$

for epoch = 1 to E **do**
  Shuffle the dataset $(\mathbf{X}, \mathbf{Y})$
  for each mini-batch $(\mathbf{X}_b, \mathbf{Y}_b)$ in $(\mathbf{X}, \mathbf{Y})$ **do**
    Compute ELBO for the mini-batch by Equation (10) and (13),
    Compute gradients $\nabla_{\phi_\lambda}\mathcal{L}$, $\nabla_{\phi_z}\mathcal{L}$, $\nabla_{\boldsymbol{m}_t}\mathcal{L}$, $\nabla_{\boldsymbol{S}_t}\mathcal{L}$
    Update parameters using gradient ascent:
      $\phi_\lambda \leftarrow \phi_\lambda + \eta \cdot \nabla_{\phi_\lambda}\mathcal{L}$
      $\phi_z \leftarrow \phi_z + \eta \cdot \nabla_{\phi_z}\mathcal{L}$
      $\boldsymbol{m}_t \leftarrow \boldsymbol{m}_t + \eta \cdot \nabla_{\boldsymbol{m}_t}\mathcal{L}$
      $\boldsymbol{S}_t \leftarrow \boldsymbol{S}_t + \eta \cdot \nabla_{\boldsymbol{S}_t}\mathcal{L}$
  end for
end for
return ϕ_λ, ϕ_z, $\boldsymbol{m}_t$, $\boldsymbol{S}_t$

for applications such as predicting agricultural commodity prices [2,3], surrogate modeling [4], Bayesian optimization [5], and reinforcement learning [6]. Despite their effectiveness, GPs suffer from a computational complexity of $\mathcal{O}(N^3)$, where N is the number of training points. This high complexity arises from the need to compute and invert the $N \times N$ covariance matrix, which becomes prohibitive for large datasets.

Sparse GPs. Sparse Gaussian Processes are an extension of GPs that address scalability issues in modeling large datasets. Traditional GPs involve inverting the covariance matrix, which becomes computationally expensive as the size of the dataset increases. Sparse GPs alleviate this issue by introducing a smaller set of "inducing points" that approximate the latent function properties over the entire dataset. By assuming a joint distribution over the function values at the inducing points and the entire data set, Sparse GPs can approximate the true GP model while significantly reducing the computational complexity. Sparse GPs have gained popularity in various applications, including machine learning [25,43], robotics [44], and computer vision [45], where dealing with large datasets is common. Modeling and inference with Sparse GPs have evolved considerably over the last few years with key contributions in the direction of scalability to virtually any number of data points and generality within automatic differentiation frameworks [46–48]. This has been possible thanks to the combination of stochastic variational inference techniques [42] with representations based on inducing variables [19,25,49]. These advancements have now made GPs attractive to a variety of applications and likelihoods [46,50–52]. However, it is worth noting that Sparse GPs still require selecting the appropriate hyperparameters and inducing points, which can be challenging in some cases.

Fully Bayesian GPs. The key distinction between fully Bayesian GPs and traditional sparse Gaussian Processes is in their approach towards kernel hyperparameters. In traditional sparse Gaussian Processes, kernel hyperparameters are considered fixed or directly optimized model parameters. This means that during the modeling process, one needs to select a set of optimal hyperparameter values to fit the training data. While this approach can yield good results when the training data is abundant and of high quality, selecting appropriate hyperparameter values can become challenging in situations with scarce or nonlinear data. Fully Bayesian GPs treat kernel hyperparameters as random variables and introduce prior distributions to represent their uncertainty. This means that Fully Bayesian GPs no longer rely on fixed hyperparameter values but instead model the range of possible hyperparameter values and update their prior distributions based on the posterior distribution from the observed data. This approach enables the estimation of the true values of hyperparameters and their uncertainties using Bayesian inference, allowing for better adaptation to the data.

Fully Bayesian Gaussian processes have been used extensively by numerous researchers. In early studies, [53,54] investigated the use of Hamiltonian Monte Carlo (HMC) methods to perform integration over covariance hyperparameters in the regression setting. [55] extended the application of HMC methods to the classification setting. They employed HMC for sampling in the hyperparameter space and utilized the Laplace approximation to compute the integral over function values. [56] focused on MCMC schemes to sample covariance hyperparameters in conjunction with latent function values, mainly mitigating the coupling effect through reparameterization. [26] considered joint sampling of inducing variables and hyperparameters from the optimal variational posterior distribution while [57] considered inference schemes for fully Bayesian sparse GPs in a streaming setting. The technique introduced by [13] incorporates Variational Inference (VI) into Fully Bayesian GPs, approximating the posterior over hyperparameters with a factorized Gaussian distribution (mean-field approximation). More recently, [58] modified the generative model by adding a prior over the inducing inputs, and performed inference using SG-HMC over the joint kernel hyperparameters λ, inducing points $\mathbf{Z}$, inducing variables $\mathbf{U}$ space. Subsequently, [14] extended this method to a doubly collapsed bound, which analytically selected the optimal distribution over the inducing points.

GPs with deep architectures. We also focus on the utilization of deep structures in GPs, specifically Deep Gaussian Processes (DGPs) and Differential Gaussian Processes (DiffGPs) consisting of discrete layers and continuous layers. DGP [7] is a model composed of multiple layers of Gaussian processes. Each layer is a Gaussian process used to model the nonlinear relationships in intermediate layers. The output layer of the DGP provides the final prediction. In addition to traditional Bayesian inference methods, DGP introduces several new inference techniques. DSVI [28] introduced stochastic variational inference to handle large-scale data and learn the distribution of model parameters. SGHMC [59] is a model that uses Hamiltonian Monte Carlo method for inference in deep Gaussian processes. This method incorporates stochastic gradients for learning, allowing for large-scale data processing during the inference process. IPVI [60] constructed an approximate posterior by introducing Nash equilibrium. NOVI [61] is a method that uses neural networks and score-based approaches to approximate the complex posterior distribution in DGPs. Unlike traditional DGP methods that focus on iterative function mappings, DiffGPs [10] utilized SDEs to characterize continuous-depth Gaussian processes. By transforming the GP modeling into a continuous-time framework and introducing SDEs to describe time evolution, DiffGPs can learn continuous-time transformations or flows of the data. Based on [10,62] derived direct approximations to the Fokker–Planck–Kolmogorov (FPK) equation in an assumed density Gaussian form that avoids sampling-based inference in the latent space, which makes inference fast. Our work builds upon the model proposed by [10] and combines it with the advantages of Fully Bayesian Gaussian processes. Additionally, we treat the hyperparameters as time-varying and utilize coupled neural SDEs for posterior inference of both the hyperparameters and inducing points. We summarize these methods in Table 1.

SDE solvers. SDE solvers are numerical methods for approximating the state trajectories of the system over time by discretizing the SDEs into steps and iteratively updating the state variables. They are widely used in scientific and engineering fields, especially in computing Neural SDEs [16,17,64]. In Neural SDEs, the drift and diffusion terms of the SDE are parameterized using neural networks. This allows for more expressive modeling of the dynamics compared to traditional SDEs. SDE solvers are vital for training and inference in Neural SDE models. By approximating state trajectories, SDE solvers enable the calculation of gradients [15,65], which is necessary for optimizing neural network parameters using techniques like stochastic gradient

Table 1
We summarize the existing literature on sparse GPs, deep GPs, and continuous-time GPs, focusing on the handling methods for kernel hyperparameters λ, inducing points $\mathbf{Z}$, inducing variables $\mathbf{U}$, and the associated inference techniques, whether through point estimation or Bayesian estimation. From this table, it is evident that our approach incorporates the idea of continuous layers and employs time-varying Bayesian posterior inference, significantly enhancing the model's flexibility and robustness.

	Method name	Sparse	Layer	Time-vary	$\mathbf{U}$	λ	$\mathbf{Z}$	Inference	Ref.
GP	GP	×	Single	–	–	Point	–	Maximum likelihood	[1]
Sparse GP	FITC-SVGP	✓	Single	–	Bayes	Point	Point	VI	[19]
	SVGP	✓	Single	–	Bayes	Point	Point	SVI	[25]
Bayesian GP	SMCMC-GP	✓	Single	–	Bayes	Bayes	Point	MCMC	[26]
	SSGP	✓	Single	–	Bayes	Bayes	Point	Streaming VI	[63]
	Bayesian GPR	✓	Single	–	Bayes	Bayes	Point	VI/HMC	[13]
	BSGP	✓	Single	–	Bayes	Bayes	Bayes	SG-HMC	[58]
	SGPR + HMC	✓	Single	–	Bayes	Bayes	Point	Collapsed VI	[14]
Deep GP	DGP	×	Discrete	–	–	Point	–	VI	[7]
	DSVI-DGP	✓	Discrete	–	Bayes	Point	Point	DSVI	[28]
	SGHMC-DGP	✓	Discrete	–	Bayes	Point	Point	SGHMC	[59]
	IPVI-DGP	✓	Discrete	–	Bayes	Point	Point	IPVI	[60]
	NOVI-DGP	✓	Discrete	–	Bayes	Point	Point	NOVI	[61]
Cont-time GP	DiffGP	✓	Continuous	×	Bayes	Point	Point	SVI	[10]
	Match FPK	✓	Continuous	×	Bayes	Point	Point	Assumed density	[62]
Ours	FB-DiffGP	✓	Continuous	✓	Bayes	Bayes	Bayes	SVI + neural SDE	–

descent. Moreover, SDE solvers are also used in the simulation and generation of data from the learned Neural SDE models. These solvers enable the generation of synthetic data that captures the complex dynamics of the modeled systems, making it possible to analyze and explore the behavior of the neural system under different conditions. The use of SDE solvers in Neural SDEs has shown remarkable results in various machine learning applications including time series forecasting [66], generative modeling [67], uncertainty quantification, and reinforcement learning [68].

Relationship between generative artificial intelligence approaches. Neural Stochastic Differential Equation (SDE) methods, particularly score-based diffusion SDE models, have been extensively applied in the Generative Artificial Intelligence (GAI) field for tasks such as image synthesis [69,70], 3D generation [71], and audio creation [72]. These approaches leverage SDE solvers to generate realistic and high-quality data by modeling complex data distributions through diffusion processes. Although both our method and GAI approaches utilize black-box SDE solvers, the underlying objectives and theoretical foundations differ significantly.

Our approach is grounded in SDE theory, specifically leveraging Girsanov's theorem and Variational Inference (VI) methods to address the hyperparameter uncertainty inherent in traditional continuous-time Gaussian processes. This allows for a more principled and theoretically sound handling of uncertainty in model parameters, enhancing the robustness and reliability of the predictive models. In contrast, generative AI primarily employs SDE solvers to generate data that closely resembles real-world samples, focusing on the quality and realism of the generated outputs rather than on resolving parameter uncertainties. Consequently, although both methodologies utilize similar computational tools, their goals and theoretical underpinnings cater to distinct aspects of machine learning and data modeling. This distinction underscores the versatility of SDE-based approaches in addressing a wide range of challenges across different domains.

5. Experiments

We evaluate the performance of our approximate inference method, FB-DiffGP, on UCI datasets for regression and classification, comparing it against state-of-the-art techniques: SVGP [25], DGP [28], and DiffGP [10]. The reasons for selecting these baseline methods are as follows: DiffGP is the primary baseline for our approach, making it the main focus of our comparison because it provides an approximate inference method for continuous deep Gaussian processes. DGP [28], another important work on deep Gaussian processes, constructs a deep network structure to effectively capture the complexity of data. As FB-DiffGP shares similarities in terms of model architecture, it serves as a meaningful point of comparison. SVGP [25] is a widely used scalable sparse variational inference method in Gaussian process regression that performs well on large-scale datasets.

The number of inducing points M is manually selected to balance accuracy and computation time using cross validation. We optimize all parameters jointly using the evidence lower bound and employ stochastic optimization with mini-batches and the Adam optimizer. Numerical solutions of SDEs are obtained using the Euler–Maruyama solver with 20 time steps. The number of steps changes if the time interval T is increased. For larger T, we would typically need to increase the number of time steps to maintain the same level of accuracy in the discretization of the SDE. This ensures that the solution remains precise and the numerical methods used are effective for longer time intervals. Other solvers, such as higher-order or adaptive methods, can also be easily implemented using Python toolkits.

Our model primarily handles classification and regression tasks, employing a simple network architecture with a few fully connected layers. The mini-batch size chosen is 10,000 and the learning rate is set to 10^{-2}. Our implementation utilizes GPyTorch [48], a Gaussian processes framework based on PyTorch. For comparison, we used a simple two-layer binarized neural network (BNN). The network includes a hidden layer with a defined number of neurons, followed by an output layer. The DNN used for comparison is the same as described in [73]. This network architecture includes several layers, with details provided in that paper. Our code is available at https://github.com/xujianscut/FB-DIFFGP.

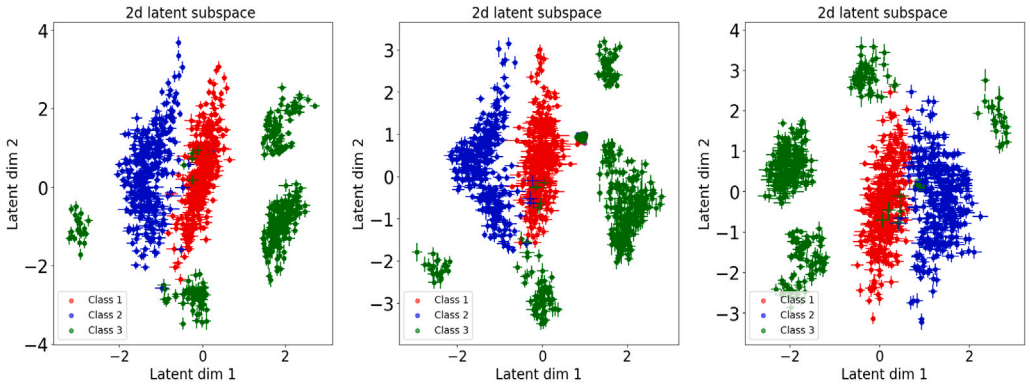
5.1. Description of datasets

Our experiments encompass both unsupervised dimensionality reduction clustering tasks and supervised Bayesian classification and regression tasks.

For unsupervised learning, presented in Section 5.2, we utilize the multi-phase Oilflow dataset [74]. This dataset consists of measurements from multi-phase oil flow processes, capturing the dynamic behavior of oil, water, and gas interactions. It is important for assessing clustering algorithms in industrial scenarios with complex flow patterns.

For supervised learning, presented in Sections 5.3 and 5.4, we employ eight benchmark regression datasets from the UCI Machine Learning Repository [75]. These datasets include:

- **Yacht Hydrodynamics**: Predicts the hydrodynamic performance of sailing yachts based on various design parameters.



(a) FB-DiffGP (Ours) (b) DiffGP (c) Standard GPLVM

Fig. 1. Unsupervised dimensionality reduction on the "Oilflow" toy data set.

- **Boston Housing**: Estimates median house prices in Boston suburbs using features such as crime rate, number of rooms, and accessibility to highways.
- **Energy Efficiency**: Models the heating and cooling load requirements of buildings based on architectural and environmental factors.
- **Concrete Compressive Strength**: Predicts the compressive strength of concrete using ingredients like cement, slag, and age.
- **Power Plant**: Forecasts the electrical power output of a power plant based on ambient variables and operating conditions.
- **Elevators**: Estimates the shaft size required for elevators in buildings based on building characteristics and elevator specifications.
- **Protein Tertiary Structure**: Predicts the spatial coordinates of protein structures from amino acid sequences.
- **Year Prediction MSD**: Forecasts the release year of songs based on audio features extracted from the Million Song Dataset.

These regression datasets vary in size, with the number of data points ranging from 308 (Yacht Hydrodynamics) to 515,345 (Year Prediction MSD). They originate from diverse real-world applications, demonstrating the scalability and applicability of our proposed method to practical regression problems across different domains.

For classification, we utilize the SUSY [76] and HIGGS [77] data sets, which are large-scale real-world datasets containing 5,500,000 and 11,000,000 samples, respectively.

- **SUSY**: This dataset involves the classification of events in high-energy physics experiments to distinguish between signal events (indicative of supersymmetry) and background noise. It is essential for evaluating the performance of classification algorithms in handling high-dimensional and large-scale data typical in particle physics.
- **HIGGS**: Designed for the classification of particle collision events to identify the production of Higgs bosons, this dataset is crucial for assessing the efficiency and operational effectiveness of classification methods in processing massive datasets prevalent in scientific research and industry applications.

Both SUSY and HIGGS datasets are derived from real-world high-energy physics experiments. They enable us to evaluate the scalability of our proposed approach. The diverse range of datasets employed in our experiments, encompassing both regression and classification tasks from various real-world domains, highlights the robustness and versatility of our FB-Diff approach.

5.2. Unsupervised learning

We applied our model to a dimensionality reduction task using the Bayesian Gaussian Process Latent Variable Model (GPLVM) [78] for data reconstruction. Our toy dataset is the multi-phase Oilflow data [74], comprising 1000 data points in 12 dimensions, categorized into three classes representing different phases of oil flow in a pipeline. We reduced the data dimensionality to 10 while retaining as much information as possible. As the training is unsupervised, the ground-truth labels were not used during training. We present the reconstruction error and mean squared error (MSE) along with ±2 standard errors from ten optimization runs. The 2D projections of the latent space for the Oilflow data clearly show that our model effectively reveals the class structure. To emphasize the strengths of our model, we present the results of the 2D latent space for three models in Fig. 1. As shown in Fig. 1, the reconstructed data points for the three classes are more distinctly separated in our model, resulting in improved and more intuitive clustering. From Table 2, we observe that our proposed FB-DiffGP outperforms both the Standard GPLVM and DiffGP methods, yielding lower reconstruction loss and improved uncertainty estimation.

Table 2
MSE and NLL for our FB-DiffGP compared to the baseline and standard GPLVM on toy "Oilflow" dataset of 1000 points in 12 dimensions.

Method	MSE	NLL
Standard GPLVM	2.45 ± 0.05	−12.42 ± 0.07
DiffGP	1.87 ± 0.04	−14.41 ± 0.04
FB-DiffGP (Ours)	1.65 ± 0.03	−16.51 ± 0.05

5.3. Regression benchmarks

We also compared our model with the state-of-the-art results from [10] on eight regression benchmarks. Each experiment used 90%/10% random training and testing splits, with 20 repetitions. For both Gaussian Process methods, we used the RBF kernel with ARD and 100 inducing points. During testing, we computed the predictive mean and variance for each sample generated from Eq. (6) and calculated the average summary statistics, including RMSE and Log Likelihood (LL), across these samples. The mean and standard error of RMSE are reported in Table 3, while the mean and standard error of LL are in Table 4. From Tables 3 and 4, it is clear that our method outperforms previous methods on all eight datasets, with significant improvements observed on the Boston, Energy, Concrete, and Protein datasets. The data were tested with flow time values ranging from 1 to 5. We also observed a similar trend as [10], where increasing the flow time can increase the model capacity without overfitting, consequently improving the model's generalization ability.

Table 3
The test RMSE values on 8 benchmark datasets using 90%/10% random training and test splits with 20 repetitions.

		Yacht	Boston	Energy	Concrete
	$N \mid D$	308 | 6	506 | 13	768 | 8	1030 | 8
Linear		0.68 ± 0.05	4.24 ± 0.16	2.88 ± 0.05	10.54 ± 0.13
BNN	$L = 2$	0.47 ± 0.04	3.01 ± 0.18	1.80 ± 0.05	5.67 ± 0.09
Sparse GP	$M = 100$	0.45 ± 0.04	2.87 ± 0.15	0.78 ± 0.02	5.97 ± 0.11
	$M = 500$	0.44 ± 0.04	2.73 ± 0.12	0.47 ± 0.02	5.53 ± 0.12
Deep GP $M = 100$	$L = 2$	0.45 ± 0.03	2.90 ± 0.17	0.47 ± 0.01	5.61 ± 0.10
	$L = 3$	0.45 ± 0.03	2.93 ± 0.16	0.48 ± 0.01	5.64 ± 0.10
	$L = 4$	0.44 ± 0.03	2.90 ± 0.15	0.48 ± 0.01	5.68 ± 0.10
	$L = 5$	0.42 ± 0.03	2.92 ± 0.17	0.47 ± 0.01	5.65 ± 0.10
DiffGP $M = 100$	$T = 1.0$	0.45 ± 0.04	2.80 ± 0.13	0.49 ± 0.02	5.32 ± 0.10
	$T = 2.0$	0.43 ± 0.04	2.68 ± 0.10	0.48 ± 0.02	4.96 ± 0.09
	$T = 3.0$	0.43 ± 0.03	2.69 ± 0.14	0.47 ± 0.02	4.76 ± 0.12
	$T = 4.0$	0.42 ± 0.03	2.67 ± 0.13	0.49 ± 0.02	4.65 ± 0.12
	$T = 5.0$	0.40 ± 0.04	2.58 ± 0.12	0.50 ± 0.02	4.56 ± 0.12
FB-DiffGP (ours) $M = 100$	$T = 1.0$	**0.43** ± 0.04	**2.63** ± 0.10	**0.42** ± 0.01	**4.75** ± 0.12
	$T = 2.0$	**0.41** ± 0.03	**2.49** ± 0.10	**0.41** ± 0.02	**4.33** ± 0.11
	$T = 3.0$	**0.41** ± 0.03	**2.47** ± 0.11	**0.39** ± 0.01	**4.22** ± 0.12
	$T = 4.0$	**0.40** ± 0.02	**2.45** ± 0.09	**0.37** ± 0.02	**4.07** ± 0.11
	$T = 5.0$	**0.38** ± 0.04	**2.39** ± 0.10	**0.38** ± 0.01	**4.01** ± 0.11
		Power	Elevators	Protein	Year
	$N \mid D$	9568 | 4	16,599 | 18	45,730 | 9	515,345 | 90
Linear		4.51 ± 0.03	5.08 ± 0.03	5.21 ± 0.02	6.35 ± 0.05
BNN	$L = 2$	4.12 ± 0.03	4.57 ± 0.03	4.73 ± 0.01	5.78 ± 0.05
Sparse GP	$M = 100$	3.91 ± 0.03	4.47 ± 0.03	4.43 ± 0.03	5.59 ± 0.06
	$M = 500$	3.79 ± 0.03	4.32 ± 0.03	4.10 ± 0.03	5.33 ± 0.04
Deep GP $M = 100$	$L = 2$	3.79 ± 0.03	4.35 ± 0.04	4.00 ± 0.03	5.43 ± 0.04
	$L = 3$	3.73 ± 0.04	4.34 ± 0.03	3.81 ± 0.04	5.38 ± 0.04
	$L = 4$	3.71 ± 0.04	4.32 ± 0.03	3.74 ± 0.04	5.25 ± 0.03
	$L = 5$	3.68 ± 0.03	4.30 ± 0.03	**3.72** ± 0.04	**5.23** ± 0.03
DiffGP $M = 100$	$T = 1.0$	3.76 ± 0.03	4.38 ± 0.03	4.04 ± 0.04	5.45 ± 0.04
	$T = 2.0$	3.72 ± 0.03	4.33 ± 0.03	4.00 ± 0.04	5.41 ± 0.03
	$T = 3.0$	3.68 ± 0.03	4.32 ± 0.03	3.92 ± 0.04	5.37 ± 0.03
	$T = 4.0$	3.66 ± 0.03	4.30 ± 0.03	3.89 ± 0.04	5.33 ± 0.03
	$T = 5.0$	3.65 ± 0.03	4.30 ± 0.02	3.87 ± 0.04	5.30 ± 0.03
FB-DiffGP (ours) $M = 100$	$T = 1.0$	**3.64** ± 0.03	**4.32** ± 0.03	**3.94** ± 0.03	**5.40** ± 0.04
	$T = 2.0$	**3.61** ± 0.03	**4.30** ± 0.03	**3.88** ± 0.03	**5.36** ± 0.03
	$T = 3.0$	**3.58** ± 0.03	**4.28** ± 0.04	**3.85** ± 0.03	**5.33** ± 0.03
	$T = 4.0$	**3.54** ± 0.03	**4.25** ± 0.03	**3.81** ± 0.03	**5.28** ± 0.03
	$T = 5.0$	**3.53** ± 0.03	**4.25** ± 0.02	**3.79** ± 0.03	**5.24** ± 0.03

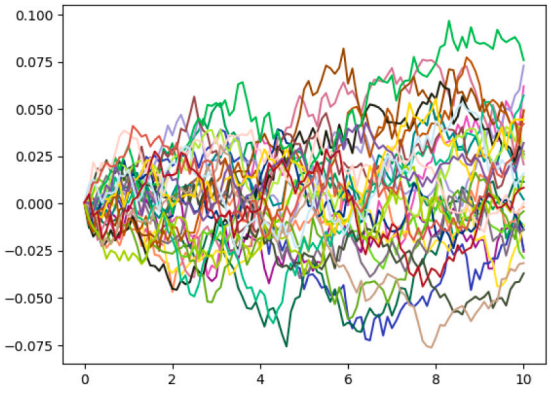


Fig. 2. We display 30 sample paths to visualize the trajectories of the SDE for the Concrete data set. The plot shows that the posterior of our kernel hyperparameters is not a point estimate but a posterior SDE flow. The diffusion process shown in this figure is multivariate, but we only displayed a slice of it in the two-dimensional plot. The x-axis represents time t, while the y-axis represents the values of the hyperparameters.

Simulation of kernel hyperparameter flows. To illustrate the improvement over the baseline more clearly, we simulated the trajectory of the learned kernel hyperparameters λ SDE. Since λ is indeed multivariate, we only displayed a slice of it. For visualization purposes, we selected the Concrete dataset and set $T = 10.0$. The results are shown in Fig. 2. The x-axis represents time t, while the y-axis represents the values of the hyperparameters. In order to facilitate the visualization of the trajectories of the SDE, we displayed 30 sample paths in the figure and selected fixed initial values for optimization purposes. From the plot, we can observe that the posterior of our kernel hyperparameters is no longer a point estimate but a posterior SDE flow. Our trajectories serve to illustrate what the posterior SDE has learned, providing insight into the uncertainty estimation for the hyperparameters. The detailed uncertainty estimates are provided in Table 4. In Fig. 3, we display the empirical distribution of these 100 paths at $T = 10$ for two data sets.

Computational efficiency. Similar to the baseline algorithms, each training iteration of FB-DiffGP involves computing the inverse covariance with a complexity of $\mathcal{O}(M^3)$. In Table 5, we compare our method with the baseline DiffGP trained for a fixed number of epochs. The experiment is repeated five times on the same fold, and the results are averaged. Each run is conducted on a dedicated instance in a cloud computing platform equipped with a single Tesla A100 GPU and an Intel Core i9-13900K CPU. The results demonstrate that FB-DiffGP does not significantly increase computational costs, thanks to the efficient parallel processing capabilities of deep learning GPUs.

Comparison with different numbers of inducing points. We compared the performance of our algorithm with $M = 100$ and $M = 500$ inducing

Table 4
The test log-likelihood values on 8 benchmark datasets using 90%/10% random training and test splits with 20 repetitions.

		Yacht	Boston	Energy	Concrete
	$N \mid D$	308 | 6	506 | 13	768 | 8	1030 | 8
Linear		−0.72 ± 0.03	−2.89 ± 0.03	−2.48 ± 0.02	−3.78 ± 0.01
BNN	$L = 2$	−0.64 ± 0.06	−2.57 ± 0.09	−2.04 ± 0.02	−3.16 ± 0.02
Sparse GP	$M = 100$	−0.61 ± 0.04	−2.47 ± 0.05	−1.29 ± 0.02	−3.18 ± 0.02
	$M = 500$	−0.57 ± 0.03	−2.40 ± 0.07	−0.93 ± 0.01	−3.09 ± 0.02
Deep GP $M = 100$	$L = 2$	−0.61 ± 0.05	−2.47 ± 0.05	−0.73 ± 0.02	−3.12 ± 0.01
	$L = 3$	−0.58 ± 0.04	−2.49 ± 0.05	−0.75 ± 0.02	−3.13 ± 0.01
	$L = 4$	−0.60 ± 0.04	−2.48 ± 0.05	−0.76 ± 0.02	−3.14 ± 0.01
	$L = 5$	−0.58 ± 0.03	−2.49 ± 0.05	−0.74 ± 0.02	−3.13 ± 0.01
DiffGP $M = 100$	$T = 1.0$	−0.57 ± 0.04	−2.36 ± 0.04	−0.65 ± 0.03	−3.05 ± 0.02
	$T = 2.0$	−0.55 ± 0.03	−2.32 ± 0.04	−0.63 ± 0.03	−2.96 ± 0.02
	$T = 3.0$	−0.53 ± 0.03	−2.31 ± 0.05	−0.63 ± 0.03	−2.93 ± 0.04
	$T = 4.0$	−0.56 ± 0.05	−2.33 ± 0.06	−0.65 ± 0.03	−2.91 ± 0.04
	$T = 5.0$	−0.54 ± 0.03	−2.30 ± 0.05	−0.66 ± 0.03	−2.90 ± 0.05
FB-DiffGP (ours) $M = 100$	$T = 1.0$	**−0.51** ± 0.03	**−2.28** ± 0.04	**−0.62** ± 0.03	**−2.75** ± 0.03
	$T = 2.0$	**−0.49** ± 0.03	**−2.26** ± 0.05	**−0.59** ± 0.03	**−2.72** ± 0.03
	$T = 3.0$	**−0.47** ± 0.03	**−2.23** ± 0.04	**−0.56** ± 0.04	**−2.63** ± 0.03
	$T = 4.0$	**−0.44** ± 0.04	**−2.21** ± 0.05	**−0.54** ± 0.03	**−2.63** ± 0.04
	$T = 5.0$	**−0.43** ± 0.03	**−2.19** ± 0.04	**−0.52** ± 0.02	**−2.60** ± 0.03
		Power	Elevators	Protein	Year
	$N \mid D$	9568 | 4	16,599 | 18	45,730 | 9	515,345 | 90
Linear		−2.93 ± 0.01	−2.76 ± 0.03	−3.07 ± 0.00	−6.32 ± 0.03
BNN	$L = 2$	−2.84 ± 0.01	−2.45 ± 0.05	−2.97 ± 0.00	−5.62 ± 0.03
Sparse GP	$M = 100$	−2.75 ± 0.01	−2.41 ± 0.04	−2.91 ± 0.00	−5.46 ± 0.03
	$M = 500$	−2.75 ± 0.01	−2.26 ± 0.04	−2.83 ± 0.00	−5.38 ± 0.03
Deep GP $M = 100$	$L = 2$	−2.75 ± 0.01	−2.32 ± 0.03	−2.81 ± 0.00	−5.48 ± 0.04
	$L = 3$	−2.74 ± 0.01	−2.28 ± 0.03	−2.75 ± 0.00	−5.42 ± 0.04
	$L = 4$	−2.74 ± 0.01	−2.25 ± 0.03	−2.73 ± 0.00	−5.36 ± 0.03
	$L = 5$	−2.73 ± 0.01	−2.24 ± 0.03	−2.71 ± 0.00	−5.34 ± 0.03
DiffGP $M = 100$	$T = 1.0$	−2.75 ± 0.01	−2.30 ± 0.03	−2.79 ± 0.04	−5.40 ± 0.04
	$T = 2.0$	−2.74 ± 0.01	−2.27 ± 0.04	−2.78 ± 0.04	−5.36 ± 0.05
	$T = 3.0$	−2.72 ± 0.01	−2.25 ± 0.04	−2.79 ± 0.00	−5.31 ± 0.04
	$T = 4.0$	−2.72 ± 0.01	−2.25 ± 0.03	−2.78 ± 0.00	−5.29 ± 0.03
	$T = 5.0$	−2.72 ± 0.01	−2.26 ± 0.02	−2.77 ± 0.00	−5.28 ± 0.03
FB-DiffGP (ours) $M = 100$	$T = 1.0$	**−2.68** ± 0.01	**−2.28** ± 0.04	**−2.59** ± 0.02	**−5.18** ± 0.04
	$T = 2.0$	**−2.66** ± 0.01	**−2.26** ± 0.04	**−2.58** ± 0.02	**−5.16** ± 0.03
	$T = 3.0$	**−2.66** ± 0.01	**−2.25** ± 0.03	**−2.56** ± 0.01	**−5.14** ± 0.03
	$T = 4.0$	**−2.63** ± 0.02	**−2.24** ± 0.04	**−2.57** ± 0.01	**−5.13** ± 0.04
	$T = 5.0$	**−2.62** ± 0.04	**−2.23** ± 0.03	**−2.55** ± 0.01	**−5.11** ± 0.03

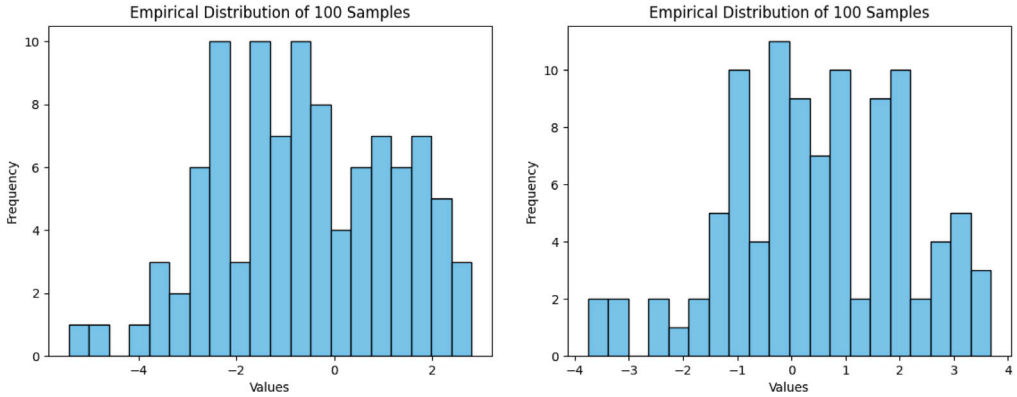


Fig. 3. An empirical histogram of 100 sample paths at time $T = 10.0$ for the Concrete (left) and Energy (right) datasets is shown. The x-axis represents the values of the kernel parameters, while the y-axis shows the empirical distribution. From the figure, it is clear that the learned posterior distribution of kernel parameters is no longer a point estimate, but instead takes the form of a probability distribution.

points, shown in Table 6. Our findings indicate that, similar to classical sparse GP methods, the performance improves with a larger number of inducing points. However, this improvement comes at the cost of increased training time, as the computational complexity grows with the number of inducing points.

Comparison with recent deep Gaussian process baselines. To further demonstrate our proposed method, we compare with the IPVI [60] and NOVI [61] models, which are recent deep Gaussian process models that also utilize neural networks for inference. We show comparisons using the UCI datasets, evaluating the models based on RMSE (Table 7).

Table 5
Runtime (in seconds) of the proposed algorithm compared to the baseline for $M = 100$ and $T = 1$. Shown is the time required to complete one full pass of the entire dataset, also known as one epoch. The results show that FB-DiffGP does not lead to a substantial increase in computational costs, owing to the efficient parallel processing capabilities of deep learning GPUs.

	Yacht	Boston	Energy	Concrete	Power	Elevators	Protein	Year
N	308	506	768	1030	9K	16K	45K	515K
D	6	13	8	8	4	18	9	90
DiffGP	0.59	0.61	0.62	0.70	2.34	3.85	11.7	120.6
FB-DiffGP	0.61	0.64	0.66	0.75	2.45	4.03	12.9	126.9

Table 6
The RMSE of the proposed algorithm with varying numbers of inducing points M is shown. As the number of inducing points increases, the performance improves, leading to lower RMSE values.

	Yacht	Boston	Energy	Concrete
$N \mid D$	308 | 6	506 | 13	768 | 8	1030 | 8
FB-DiffGP $M = 100$	0.43 ± 0.04	2.63 ± 0.6	0.42 ± 0.01	4.75 ± 0.12
FB-DiffGP $M = 500$	0.42 ± 0.04	2.52 ± 0.07	0.40 ± 0.01	4.46 ± 0.14
	Power	Elevators	Protein	Year
$N \mid D$	9568 | 4	16,599 | 18	45,730 | 9	515,345 | 90
FB-DiffGP $M = 100$	3.64 ± 0.03	4.32 ± 0.03	3.94 ± 0.03	5.40 ± 0.04
FB-DiffGP $M = 500$	3.62 ± 0.03	4.29 ± 0.03	3.86 ± 0.03	5.38 ± 0.03

Table 7
The test RMSE values on 8 benchmark datasets, using 90%/10% random training and test splits with 20 repetitions, compared with recent deep Gaussian process baselines.

		Yacht	Boston	Energy	Concrete
	$N \mid D$	308 | 6	506 | 13	768 | 8	1030 | 8
IPVI $M = 100$	$L = 2$	0.43 ± 0.04	2.95 ± 0.15	0.67 ± 0.03	5.32 ± 0.15
	$L = 3$	0.43 ± 0.04	2.94 ± 0.14	0.65 ± 0.04	5.28 ± 0.14
	$L = 4$	0.44 ± 0.03	2.92 ± 0.13	0.64 ± 0.02	5.29 ± 0.12
	$L = 5$	0.42 ± 0.02	2.90 ± 0.13	0.63 ± 0.02	5.27 ± 0.11
NOVI $M = 100$	$L = 2$	0.43 ± 0.03	2.65 ± 0.14	0.48 ± 0.03	4.83 ± 0.13
	$L = 3$	0.46 ± 0.03	2.70 ± 0.14	0.46 ± 0.04	4.85 ± 0.12
	$L = 4$	0.45 ± 0.04	2.76 ± 0.13	0.44 ± 0.02	4.80 ± 0.12
	$L = 5$	0.43 ± 0.03	2.74 ± 0.11	0.42 ± 0.01	4.78 ± 0.09
FB-DiffGP (ours) $M = 100$	$T = 1.0$	0.43 ± 0.04	2.63 ± 0.10	0.42 ± 0.01	4.75 ± 0.12
	$T = 2.0$	0.41 ± 0.03	2.49 ± 0.10	0.41 ± 0.02	4.33 ± 0.11
	$T = 3.0$	0.41 ± 0.03	2.47 ± 0.11	0.39 ± 0.01	4.22 ± 0.12
	$T = 4.0$	0.40 ± 0.02	2.45 ± 0.09	0.37 ± 0.02	4.07 ± 0.11
	$T = 5.0$	**0.38** ± 0.04	**2.39** ± 0.10	**0.38** ± 0.01	**4.01** ± 0.11
	$N \mid D$	Power	Elevators	Protein	Year
		9568 | 4	16,599 | 18	45,730 | 9	515,345 | 90
IPVI $M = 100$	$L = 2$	3.78 ± 0.03	4.38 ± 0.04	3.98 ± 0.03	5.44 ± 0.04
	$L = 3$	3.74 ± 0.03	4.35 ± 0.03	3.77 ± 0.04	5.38 ± 0.04
	$L = 4$	3.70 ± 0.03	4.32 ± 0.03	3.75 ± 0.04	5.34 ± 0.03
	$L = 5$	3.68 ± 0.03	4.30 ± 0.03	3.72 ± 0.03	5.31 ± 0.02
NOVI $M = 100$	$L = 2$	3.79 ± 0.03	4.34 ± 0.04	3.95 ± 0.03	5.42 ± 0.04
	$L = 3$	3.75 ± 0.04	4.33 ± 0.04	3.81 ± 0.03	5.38 ± 0.03
	$L = 4$	3.73 ± 0.03	4.31 ± 0.03	3.74 ± 0.03	5.32 ± 0.03
	$L = 5$	3.70 ± 0.03	4.29 ± 0.02	**3.71** ± 0.03	5.28 ± 0.03
FB-DiffGP (ours) $M = 100$	$T = 1.0$	3.64 ± 0.03	4.32 ± 0.03	3.94 ± 0.03	5.40 ± 0.04
	$T = 2.0$	3.61 ± 0.03	4.30 ± 0.03	3.88 ± 0.03	5.36 ± 0.03
	$T = 3.0$	3.58 ± 0.03	4.28 ± 0.04	3.85 ± 0.03	5.33 ± 0.03
	$T = 4.0$	3.54 ± 0.03	4.25 ± 0.03	3.81 ± 0.03	5.28 ± 0.03
	$T = 5.0$	**3.53** ± 0.03	**4.25** ± 0.02	3.79 ± 0.03	**5.24** ± 0.03

The results show that our FB-DiffGP method remains competitive when compared to these recent state-of-the-art methods.

5.4. Classification benchmarks

We performed large-scale experiments using the Higgs dataset, which contains 11 million data points with 28 features. This dataset was created through Monte Carlo simulations modeling particle dynamics in accelerators for Higgs boson detection. We randomly split the data, using 90% for training and the remaining 10% for testing. To evaluate the performance, we use the area under the curve (AUC) metric and compared our results with previously reported methods. Table 8 presents the obtained test performance, demonstrating that our FB-DiffGP method outperforms the competing approaches. Additionally, we conducted experiments on the SUSY dataset, and the results showcased the competitive performance of our proposed algorithm.

6. Conclusion and future work

We have proposed a fully Bayesian approach to Gaussian Process (GP) modeling, where kernel hyperparameters are treated as random variables, and interconnected stochastic differential equations are used to infer the posterior distribution of DiffGPs. By capturing uncertainty in hyperparameter estimation, our method significantly enhances the

Table 8
The test AUC values for large-scale classification datasets, using a 90%/10% random split for training and testing, show that our method overall outperforms the baseline, further demonstrating the scalability of the approach.

		SUSY	HIGGS
	$N \mid D$	5,500,000 | 18	11,000,000 | 28
DNN		0.876	**0.885**
Sparse GP	$M = 100$	0.875	0.785
	$M = 500$	0.876	0.794
Deep GP $M = 100$	$L = 2$	0.877	0.830
	$L = 3$	0.877	0.837
	$L = 4$	0.877	0.841
	$L = 5$	0.877	0.846
DiffGP $M = 100$	$T = 1.0$	0.878	0.840
	$T = 3.0$	0.878	0.841
	$T = 5.0$	0.878	0.842
FB-DiffGP (ours) $M = 100$	$T = 1.0$	**0.887**	0.852
	$T = 3.0$	**0.887**	0.856
	$T = 5.0$	**0.887**	0.857

model's adaptability to complex system dynamics. Experimental results demonstrate that our approach outperforms conventional techniques, showing improved accuracy.

However, there are limitations to our method, including the optimization of hyperparameters such as the SDE time T, the number of inducing points M, and the architecture of the neural SDE. These require additional methods, such as cross-validation or Bayesian optimization, to tune effectively. In our current experiments, we primarily relied on empirical methods. Furthermore, balancing the trade-off between accuracy and computational efficiency remains an open challenge, which we plan to address in future work. This research also can generalize to other problems involving continuous-time Gaussian processes, e.g., [79]. Future exploration includes expanding the application of our FB-DiffGP model in diverse domains such as spatio-temporal model averaging, image analysis, and financial datasets.

CRediT authorship contribution statement

Jian Xu: Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation. **Zhiqi Lin:** Writing – review & editing. **Min Chen:** Writing – review & editing. **Junmei Yang:** Writing – review & editing. **Delu Zeng:** Writing – review & editing, Software, Conceptualization. **John Paisley:** Writing – review & editing.

Ethical and informed consent for data used

Our study followed ethical guidelines and obtained informed consent from all participants regarding the data used. Additionally, we confirm that this manuscript has not been submitted to or published by any other publication.

Declaration of competing interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Data availability

Data will be made available on request.

References

[1] C.E. Rasmussen, Gaussian processes in machine learning, in: Summer School on Machine Learning, Springer, 2003, pp. 63–71.
[2] B. Jin, X. Xu, Forecasting wholesale prices of yellow corn through the Gaussian process regression, Neural Comput. Appl. 36 (15) (2024) 8693–8710.
[3] B. Jin, X. Xu, Machine learning coffee price predictions, J. Uncertain Syst. 17 (04) (2024) 2450023.
[4] A. Marrel, B. Iooss, Probabilistic surrogate modeling by Gaussian process: A review on recent insights in estimation and validation, Reliab. Eng. Syst. Saf. (2024) 110094.
[5] Z. Wang, G.E. Dahl, K. Swersky, C. Lee, Z. Nado, J. Gilmer, J. Snoek, Z. Ghahramani, Pre-trained Gaussian processes for Bayesian optimization, J. Mach. Learn. Res. 25 (212) (2024) 1–83.
[6] W. Zhao, T. He, C. Liu, Probabilistic safeguard for reinforcement learning using safety index guided Gaussian process models, in: Learning for Dynamics and Control Conference, PMLR, 2023, pp. 783–796.
[7] A. Damianou, N.D. Lawrence, Deep Gaussian processes, in: Artificial Intelligence and Statistics, 2013.
[8] D. Duvenaud, O. Rippel, R. Adams, Z. Ghahramani, Avoiding pathologies in very deep networks, in: Artificial Intelligence and Statistics, 2014.
[9] M.M. Dunlop, M.A. Girolami, A.M. Stuart, A.L. Teckentrup, How deep are deep Gaussian processes? J. Mach. Learn. Res. 19 (54) (2018) 1–46.
[10] P. Hegde, M. Heinonen, H. Lähdesmäki, S. Kaski, Deep learning with differential Gaussian process flows, 2018, arXiv preprint arXiv:1810.04066.
[11] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[12] R.T. Chen, Y. Rubanova, J. Bettencourt, D.K. Duvenaud, Neural ordinary differential equations, Adv. Neural Inf. Process. Syst. 31 (2018).
[13] V. Lalchand, C.E. Rasmussen, Approximate inference for fully Bayesian Gaussian process regression, in: Symposium on Advances in Approximate Bayesian Inference, 2020.
[14] V. Lalchand, W. Bruinsma, D. Burt, C.E. Rasmussen, Sparse Gaussian process hyperparameters: Optimize or integrate? in: Advances in Neural Information Processing Systems, 2022.
[15] X. Li, T.-K.L. Wong, R.T. Chen, D. Duvenaud, Scalable gradients for stochastic differential equations, in: Artificial Intelligence and Statistics, 2020.
[16] X. Liu, T. Xiao, S. Si, Q. Cao, S. Kumar, C.-J. Hsieh, How does noise help robustness? Explanation and exploration under the neural SDE framework, in: Computer Vision and Pattern Recognition, 2020.
[17] L. Kong, J. Sun, C. Zhang, SDE-NET: Equipping deep neural networks with uncertainty estimates, 2020, arXiv preprint arXiv:2008.10546.
[18] B. Tzen, M. Raginsky, Neural stochastic differential equations: Deep latent gaussian models in the diffusion limit, 2019, arXiv preprint arXiv:1905.09883.
[19] M. Titsias, Variational learning of inducing variables in sparse Gaussian processes, in: Artificial Intelligence and Statistics, 2009.
[20] J. Paisley, X. Liao, L. Carin, Active learning and basis selection for kernel-based linear models: A Bayesian perspective, IEEE Trans. Signal Process. 58 (5) (2010) 2686–2700.
[21] D. Liang, J. Paisley, Landmarking manifolds with Gaussian processes, in: International Conference on Machine Learning, 2015.
[22] J. Xu, D. Zeng, J. Paisley, Sparse inducing points in deep Gaussian processes: Enhancing modeling with denoising diffusion variational inference, in: International Conference on Machine Learning, 2024.
[23] G.-L. Tran, D. Milios, P. Michiardi, M. Filippone, Sparse within sparse gaussian processes using neighbor information, in: International Conference on Machine Learning, PMLR, 2021, pp. 10369–10378.
[24] B. Jafrasteh, C. Villacampa-Calvo, D. Hernández-Lobato, Input dependent sparse gaussian processes, 2021, arXiv preprint arXiv:2107.07281.
[25] J. Hensman, A. Matthews, Z. Ghahramani, Scalable variational Gaussian process classification, in: Artificial Intelligence and Statistics, 2015.
[26] J. Hensman, A.G. Matthews, M. Filippone, Z. Ghahramani, MCMC for variationally sparse Gaussian processes, in: Advances in Neural Information Processing Systems, 2015.
[27] C. Yildiz, M. Heinonen, J. Intosalmi, H. Mannerstrom, H. Lahdesmaki, Learning stochastic differential equations with Gaussian processes without gradient matching, in: Workshop on Machine Learning for Signal Processing, 2018.
[28] H. Salimbeni, M. Deisenroth, Doubly stochastic variational inference for deep Gaussian processes, Adv. Neural Inf. Process. Syst. (2017).
[29] T. Zhang, Z. Yao, A. Gholami, J.E. Gonzalez, K. Keutzer, M.W. Mahoney, G. Biros, ANODEV2: A coupled neural ODE framework, in: Advances in Neural Information Processing Systems, 2019.
[30] K.O. Stanley, D.B. D'Ambrosio, J. Gauci, A hypercube-based encoding for evolving large-scale neural networks, Artif. Life 15 (2) (2009) 185–212.
[31] J. Koutnik, F. Gomez, J. Schmidhuber, Evolving neural networks in compressed weight space, in: Conference on Genetic and Evolutionary Computation, 2010.
[32] D. Ha, A.M. Dai, Q.V. Le, Hypernetworks, 2016, CoRR abs/1609.09106. arXiv:1609.09106. URL http://arxiv.org/abs/1609.09106.

[33] Y. Kim, S. Wiseman, A. Miller, D. Sontag, A. Rush, Semi-amortized variational autoencoders, in: International Conference on Machine Learning, 2018.
[34] A. Agrawal, J. Domke, Amortized variational inference for simple hierarchical models, in: Advances in Neural Information Processing Systems, 2021.
[35] M. Opper, Variational inference for stochastic differential equations, Ann. Phys., Lpz. 531 (3) (2019).
[36] P. Kidger, J. Foster, X. Li, T.J. Lyons, Neural SDEs as infinite-dimensional GANs, in: International Conference on Machine Learning, 2021.
[37] W. Xu, R.T. Chen, X. Li, D. Duvenaud, Infinitely deep Bayesian neural networks with stochastic differential equations, in: Artificial Intelligence and Statistics, 2022.
[38] N.G. Van Kampen, Stochastic differential equations, Phys. Rep. 24 (3) (1976) 171–228.
[39] M. Boué, P. Dupuis, A variational representation for certain functionals of Brownian motion, Ann. Probab. 26 (4) (1998) 1641–1659.
[40] D.J. Rezende, S. Mohamed, D. Wierstra, Stochastic backpropagation and approximate inference in deep generative models, in: International Conference on Machine Learning, 2014.
[41] S.-i. Amari, Backpropagation and stochastic gradient descent method, Neurocomputing 5 (4–5) (1993) 185–196.
[42] M.D. Hoffman, D.M. Blei, C. Wang, J. Paisley, Stochastic variational inference, J. Mach. Learn. Res. (14) (2013) 1303–1347.
[43] S. Sun, J. Paisley, Q. Liu, Location dependent Dirichlet processes, in: Conference on Intelligence Science and Big Data Engineering, 2017.
[44] J. Schreiter, P. Englert, D. Nguyen-Tuong, M. Toussaint, Sparse Gaussian process regression for compliant, real-time robot control, in: IEEE International Conference on Robotics and Automation, 2015.
[45] G.-L. Tran, E.V. Bonilla, J. Cunningham, P. Michiardi, M. Filippone, Calibrating deep convolutional Gaussian processes, in: Artificial Intelligence and Statistics, 2019.
[46] A.G.d.G. Matthews, M. Van Der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, J. Hensman, GPflow: A Gaussian process library using TensorFlow, J. Mach. Learn. Res. 18 (40) (2017) 1–6.
[47] K. Krauth, E.V. Bonilla, K. Cutajar, M. Filippone, AutoGP: Exploring the capabilities and limitations of Gaussian process models, 2016, arXiv preprint arXiv:1610.05392.
[48] J. Gardner, G. Pleiss, K.Q. Weinberger, D. Bindel, A.G. Wilson, Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration, in: Advances in Neural Information Processing Systems, 2018.
[49] M. Lázaro-Gredilla, A. Figueiras-Vidal, Inter-domain Gaussian processes for sparse inference using inducing features, in: Advances in Neural Information Processing Systems, 2009.
[50] M. Van der Wilk, C.E. Rasmussen, J. Hensman, Convolutional Gaussian processes, in: Advances in Neural Information Processing Systems, 2017.
[51] P. Li, S. Chen, A review on Gaussian process latent variable models, CAAI Trans. Intell. Technol. 1 (4) (2016) 366–376.
[52] G. Corani, A. Benavoli, M. Zaffalon, Time series forecasting with Gaussian processes needs priors, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2021.
[53] J. Bernardo, J. Berger, A. Dawid, A. Smith, et al., Regression and classification using Gaussian process priors, Bayesian Stat. 6 (1998) 475.
[54] C. Williams, C. Rasmussen, Gaussian processes for regression, in: Advances in Neural Information Processing Systems, 1995.
[55] D. Barber, C. Williams, Gaussian processes for Bayesian classification via hybrid Monte Carlo, in: Advances in Neural Information Processing Systems, 1996.
[56] I. Murray, R.P. Adams, Slice sampling covariance hyperparameters of latent Gaussian models, in: Advances in Neural Information Processing Systems, 2010.
[57] T.D. Bui, C. Nguyen, R.E. Turner, Streaming sparse Gaussian process approximations, in: Advances in Neural Information Processing Systems, 2017.
[58] S. Rossi, M. Heinonen, E. Bonilla, Z. Shen, M. Filippone, Sparse Gaussian processes revisited: Bayesian approaches to inducing-variable approximations, in: Artificial Intelligence and Statistics, 2021.
[59] M. Havasi, J.M. Hernández-Lobato, J.J. Murillo-Fuentes, Inference in deep Gaussian processes using stochastic gradient Hamiltonian Monte Carlo, in: Advances in Neural Information Processing Systems, 2018.
[60] H. Yu, Y. Chen, B.K.H. Low, P. Jaillet, Z. Dai, Implicit posterior variational inference for deep Gaussian processes, in: Advances in Neural Information Processing Systems, 2019.
[61] J. Xu, S. Du, J. Yang, Q. Ma, D. Zeng, Neural operator variational inference based on regularized stein discrepancy for deep Gaussian processes, 2023, arXiv preprint arXiv:2309.12658.
[62] A. Solin, E. Tamir, P. Verma, Scalable inference in SDEs by direct matching of the Fokker–Planck–Kolmogorov equation, in: Advances in Neural Information Processing Systems, 2021.
[63] T.D. Bui, C.V. Nguyen, S. Swaroop, R.E. Turner, Partitioned variational inference: A unified framework encompassing federated and continual learning, 2018, arXiv preprint arXiv:1811.11206.
[64] X. Liu, T. Xiao, S. Si, Q. Cao, S. Kumar, C.-J. Hsieh, Neural SDE: Stabilizing neural ode networks with stochastic noise, 2019, arXiv preprint arXiv:1906.02355.
[65] P. Kidger, J. Foster, X.C. Li, T. Lyons, Efficient and accurate gradients for neural SDEs, in: Advances in Neural Information Processing Systems, 2021.
[66] L. Yang, T. Gao, Y. Lu, J. Duan, T. Liu, Neural network stochastic differential equation models with applications to financial data forecasting, Appl. Math. Model. 115 (2023) 279–299.
[67] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, in: Advances in Neural Information Processing Systems, 2020.
[68] T. Chen, L. Cheng, Y. Liu, W. Jia, S. Ma, Incremental reinforcement learning—a new continuous reinforcement learning frame based on stochastic differential equation methods, 2019, arXiv preprint arXiv:1908.02974.
[69] Y. Song, J. Sohl-Dickstein, D.P. Kingma, A. Kumar, S. Ermon, B. Poole, Score-based generative modeling through stochastic differential equations, 2020, arXiv preprint arXiv:2011.13456.
[70] H. Ma, L. Zhang, X. Zhu, J. Zhang, J. Feng, Accelerating score-based generative models for high-resolution image synthesis, 2022, arXiv preprint arXiv:2206.04029.
[71] S. Hong, D. Ahn, S. Kim, Debiasing scores and prompts of 2d diffusion for view-consistent text-to-3d generation, Adv. Neural Inf. Process. Syst. 36 (2023) 11970–11987.
[72] S. Rouard, G. Hadjeres, CRASH: Raw audio score-based generative modeling for controllable high-resolution drum sound synthesis, 2021, arXiv preprint arXiv:2106.07431.
[73] P. Baldi, P. Sadowski, D. Whiteson, Searching for exotic particles in high-energy physics with deep learning, Nat. Commun. 5 (1) (2014) 4308.
[74] C.M. Bishop, G.D. James, Analysis of multiphase flows using dual-energy gamma densitometry and neural networks, Nucl. Instrum. Methods Phys. Res. Sect. A: Accel. Spectrometers Detect. Assoc. Equip. 327 (2–3) (1993) 580–593.
[75] A. Asuncion, D. Newman, et al., UCI machine learning repository, 2007.
[76] D. Whiteson, SUSY, UCI Machine Learning Repository, 2014, http://dx.doi.org/10.24432/C54606.
[77] D. Whiteson, HIGGS, UCI Machine Learning Repository, 2014, http://dx.doi.org/10.24432/C5V312.
[78] M. Titsias, N.D. Lawrence, Bayesian Gaussian process latent variable model, in: Artificial Intelligence and Statistics, 2010.
[79] J. Paisley, S. Rowland, J.Z. Liu, B. Coull, M.-A. Kioumourtzoglou, Bayesian nonparametric model averaging using scalable Gaussian process representations, in: IEEE International Conference on Big Data, 2022.