IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 27 July 2024, accepted 12 August 2024, date of publication 14 August 2024, date of current version 26 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3443646

RESEARCH ARTICLE

A Bayesian Gaussian Process-Based Latent Discriminative Generative Decoder (LDGD) Model for High-Dimensional Data

NAVID ZIAEI[1], BEHZAD NAZARI[1], URI T. EDEN[2], ALIK S. WIDGE[3], AND ALI YOUSEFI[4]
[1]Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran
[2]Department of Mathematics and Statistics, Boston University, Boston, MA 02215, USA
[3]Department of Psychiatry and Behavioral Sciences, University of Minnesota, Minneapolis, MN 55455, USA
[4]Department of Biomedical Engineering, University of Houston, Houston, TX 77004, USA

Corresponding author: Behzad Nazari (nazari@iut.ac.ir)

ABSTRACT Extracting meaningful information from high-dimensional data poses a formidable modeling challenge, particularly when the data is obscured by noise or represented through different modalities. This research proposes a novel non-parametric modeling approach, leveraging the Gaussian process (GP), to characterize high-dimensional data by mapping it to a latent low-dimensional manifold. This model, named the latent discriminative generative decoder (LDGD), employs both the data and associated labels in the manifold discovery process. A Bayesian solution is derived to infer the latent variables, allowing LDGD to effectively capture inherent stochasticity in the data. Applications of LDGD are demonstrated on both synthetic and benchmark datasets. Not only does LDGD infer the manifold accurately, but its accuracy in predicting data points' labels surpasses state-of-the-art approaches. In the development of LDGD, inducing points are incorporated to reduce the computational complexity of Gaussian processes for large datasets, enabling batch training for enhanced efficient processing and scalability. Additionally, we show that LDGD can robustly infer manifold and precisely predict labels for scenarios in which data size is limited, demonstrating its capability to characterize high-dimensional data with limited samples efficiently. These collective attributes highlight the importance of developing non-parametric modeling approaches to analyze high-dimensional data.

INDEX TERMS High-dimensional data analysis, Gaussian process models, latent variable, variational inference, decoding.

I. INTRODUCTION

Dealing with high-dimensional data presents significant challenges, often resulting in analytical complications and increased computational demands. Dimensionality reduction techniques aim to transform complex, high-dimensional datasets into more manageable and insightful lower-dimensional representations [1]. These techniques enhance data understanding and interpretation across various scientific domains, including but not limited to neuroscience, finance, and biology [2], [3], [4], [5]. For instance, in neural data analysis, dimensionality reduction simplifies the vast amount of data from brain imaging studies or neural recordings, aiding in the identification of patterns of activity underlying cognitive processes or disorders [6], [7], [8].

Dimensionality reduction techniques fall into two main categories: parametric and non-parametric. Non-probabilistic methods include principal component analysis (PCA) [9], linear discriminant analysis (LDA) [10], and more recent approaches including t-distributed stochastic neighbor embedding (t-SNE) [11] and uniform manifold approximation and projection (UMAP) [12]. These methods are widely used for data simplification and visualization. Despite their significance in advancing science and their popularity, they

The associate editor coordinating the review of this manuscript and approving it for publication was Sawyer Duane Campbell.

often lack the ability to provide a sophisticated understanding, such as stochasticity in the inferred projections, and are highly sensitive to noise and outlier data points. On the other hand, probabilistic models, including probabilistic PCA (PPCA) [13], variational auto-encoders (VAE) [14], and Gaussian process latent variable model (GPLVM) [15] are gaining prominence as dominant approaches in data analysis. The GPLVM and PPCA address major shortcomings of non-probabilistic methods, as their inference of low-dimension representation is probabilistic rather than deterministic. GPLVM is built upon PPCA, with the advantage of utilizing non-linear mappings to find lower-dimensional representations. Despite significant progress in these models, challenges persist, and opportunities exist to enhance their ability to characterize complex and high-dimensional data. A key area for improvement is the integration of label and category information, which these models often overlook due to their predominantly unsupervised nature. An important challenge facing Gaussian process-based models like GPLVM is their excessive computational cost, especially as the data size increases. Our research seeks to tackle these computational challenges and employ label and category information to enhance the scalability and utility of models like GPLVM.

To address the high computational cost associated with non-parametric models, several approaches have been developed. The Nyström approximation [16] offered an initial method to reduce computational requirements by sampling a subset of the data to approximate the Gaussian process covariance matrix. This was followed by the introduction of the pseudo-inputs concept by [17], which parameterizes its covariance using the locations of M pseudo-input points, determined optimally through gradient-based optimization. Building upon these foundations, [18] introduced variational inducing points, utilizing variational inference for the optimal selection of inducing points. Furthermore, stochastic gradient descent-based variational inducing points, developed by [19], employed stochastic optimization to further enhance scalability for large datasets. These advancements lay the foundation for further enhancements in this category of models. For example, a method for human action recognition using an improved sparse Gaussian process latent variable model and hidden conditional random field has achieved better feature dimensionality reduction and an average recognition accuracy of 93.68% [20].

Alongside advancements in enhancing the scalability of GPLVMs, there is growing research aimed at integrating labels or categories into the dimensionality reduction process. This integration allows for the inference of both the underlying structure and its relevance within the mapping. Incorporating the labels of data points into the dimensionality reduction process has been explored in approaches such as supervised PPCA (SPPCA) [21]. This approach employs two linear mappings: one mapping from the latent space to the input space, representing the original high-dimensional data, and another mapping from the latent space to the output space, corresponding to the data points' label or category or other supervised information. This concept was extended to nonlinear dimensionality reduction techniques, especially with the GPLVM approach. The discriminative GPLVM (D-GPLVM) applies principles of generalized discriminant analysis (GDA) [22] to modify latent positions within the data, aiming to bring data points of the same class closer and distance those of differing classes. Here, latent positions denote the model's estimated, unobserved variables that capture the underlying structure of the data. Other approaches, such as the supervised GPLVM and shared GPLVM (SGPLVM) [23], [24], [25], employ two distinct GPLVMs to uncover nonlinear relationships among latent variables, data, and labels. The integration of SGPLVM with SPPCA, also known as supervised latent linear GPLVM (SLLGPLVM), was presented by [25]. This approach employs two mapping functions to link the input and output spaces via a latent variable space. To minimize computational costs, SLLGPLVM employs a linear mapping from latent variables to high-dimensional data, similar to the method used by [26]. While this modification reduces the computational cost, it sacrifices the flexibility to capture complex nonlinear relationships inherent in the data, potentially limiting the model's ability to model highly intricate patterns found in more complex datasets properly. A further extension to SGPLVM is the supervised GPLVM based on a Gaussian mixture model (SGPLVM-GMM) [26]. SGPLVM-GMM assumes that latent variables follow a Gaussian mixture distribution, where mixture components are conditioned on class labels. Thus, the model simultaneously learns to reduce data dimension and classify data samples by predicting class probabilities. These models, including SGPLVM and its extensions such as SLLGPLVM and SGPLVM-GMM, have successfully utilized latent variables and class labels to enhance feature extraction and data classification. However, they primarily rely on point estimates of the latent process, thereby increasing the risk of overfitting, especially in scenarios where linear relationships do not fully capture data structure. The semi-supervised multimodal Gaussian process latent variable model with pseudo-labels (semi-MGPPL) [27] addresses interest level estimation by integrating features from multiple modalities, although it faces challenges in obtaining behavior features for new test samples and requires all modalities for latent variable calculation.

Although these models have made progress in incorporating training labels into latent process inferencing, a gap remains in their ability to manage the complexities introduced by noise and limited sample sizes. To address these challenges, our research aims to develop a fully Bayesian dimensionality reduction solution that integrates the samples' labels. Another challenge with previous GP-based models is scalability, which this work addresses through variational inducing points. The distinctiveness of our model lies in employing a Bayesian approach for both training and inference while assuming a prior over latent space. The Bayesian approach helps us better manage data complexity introduced by noise and missing points; it also lets the framework be applied to datasets with a limited sample size. Our approach, which can be considered an extension of SGPLVM, presents a robust and scalable method for dimensionality reduction as we incorporate the inducing points. Through multiple examples, it is demonstrated that the proposed method enhances the feature extraction process by mapping data points to a lower dimension and ensures the retention of essential data patterns. We illustrate that our proposed approach can be used as a robust decoder model to draw low-dimensional representations and labels for new data points. In deriving the model, Gaussian processes (GPs) and Bayesian inference are used, turning the model into a non-parametric approach. As such, the solution prevents overfitting and is best suited for datasets with a limited number of samples. Our proposed approach can be readily applied to datasets from various fields. As such, the focus is on highlighting the framework application and evaluating its performance through synthetic and benchmark datasets.

In this research, we introduce our method, which is called Latent Discriminative Generative Decoder (or, briefly, LDGD) model. In LDGD, a low-dimensional random variable defines the underlying generative process responsible for generating both the continuous values and label data. This random variable defines the covariance structures of two Gaussian processes (GPs)—one for the continuous measurement and another for the label. A Bayesian inference solution integrated with the variational inducing points approach is employed to train the model. The usage of inducing points enables efficient management of computational costs during training and inference. Specifically, we draw two sets of inducing points, one set for each GP. LDGD achieves a balanced prediction accuracy and data generation capability with two sets of inducing points. Additionally, a batch training pipeline for LDGD is introduced, which significantly aids in faster training and enhances the model's scalability for larger datasets. LDGD also serves as an adaptive feature extraction and classification solution, allowing for predicting data point labels from continuous values. Moreover, it functions as a generative model capable of producing data in high-dimensional space. To facilitate further research and development, the code for LDGD implementation, along with the experiment files and datasets used in this study, is publicly available at the following GitHub repository: https://github.com/Navid-Ziaei/LDGD.

While some of these attributes are shared with variational auto-encoders (VAE), we argue that this framework represents a substantial advancement over VAE. A key enhancement of LDGD over VAEs is its fully Bayesian inference process, which quantifies prediction uncertainty as the posterior variance. Another notable advancement is LDGD's ability to partially optimize the dimensions of latent variables during the training phase, enhancing both feature extraction and model interpretability. Additionally, LDGD's Bayesian framework inherently protects against overfitting, making it well-suited for complex datasets with limited sample sizes. These advantages collectively position LDGD not just as a mere alternative but as a significant advancement over traditional encoder-decoder models, especially in high-dimensional data analysis.

The subsequent sections present the technical and applied aspects of the LDGD modeling framework. First, the intricacies of the LDGD framework are delved into, covering its formulation, model training, and inference procedures in Section II. Additionally, a variant of LDGD, termed "fast LDGD," which integrates a neural network into the LDGD framework to enhance its computational efficiency and potentially predictive power, is introduced. In Section III, we discuss the performance and attributes of the proposed model through evaluations on a synthetic dataset. Its applicability is then demonstrated across several benchmark datasets, including Oil Flow, Iris, and MNIST. The distinctive features of LDGD are compared with other methods, such as SLL-GPLVM, SGPLVM, and variational auto-encoder model to better understand its unique advantages and improvements of our proposed framework. Section IV discusses the results obtained in Section III addresses the potential significance of the LDGD model in efficiently reducing the dimensionality of data and its strong expressive power. Finally, Section V concludes the article with a summary of the findings and potential future research directions.

II. MATERIALS AND METHODS

A. GAUSSIAN PROCESS LATENT VARIABLE MODEL

Gaussian processes (GP) are a powerful tool in machine learning for defining distributions over functions, where function values at any finite set of points follow a multivariate normal distribution [28]. This makes GPs suitable for regression tasks, leading to Gaussian Process Regression (GPR), which infers the function that best fits the observed data with uncertainty measures [29]. Inducing points are introduced to summarize the dataset information to handle large datasets, reducing inference complexity [18]. They allow scalable GP models by approximating the full covariance matrix with a lower-rank matrix based on these inducing points. The connection between Gaussian processes, GPR, and inducing points is key to efficiently applying GPs to real-world data. These concepts are detailed in Appendix A.

Gaussian process latent variable models emerge from the necessity of understanding high-dimensional data by projecting it into a simpler and much lower-dimensional latent space. At its core, a GPLVM is built upon the hypothesis that even though data $\mathbf{Y} \in \mathbb{R}^{N\times D}$ are observed, there exists a latent representation $\mathbf{X} \in \mathbb{R}^{N\times Q}$, with $Q \ll D$, that captures the essential features or structures present in the high-dimensional data. This idea was inspired by PPCA, evolving into a nonlinear dimensionality reduction method that extends the capabilities of its precursor [13].

The relationship between GPLVM and PPCA is explained in Appendix B. In GPLVM, the conditional distribution of $\mathbf{Y}$ given $\mathbf{X}$ is assumed to be:

$$p(\mathbf{Y} \mid \mathbf{X}) = \prod_{d=1}^{D} p(\mathbf{y}_{:,d} \mid \mathbf{X}), \tag{1}$$

where $\mathbf{y}_{:,d}$ is dth column of $\mathbf{Y}$. Each element of $\mathbf{y}_{:,d}$ is a noisy realization of the function $\mathbf{f}_d \in \mathbb{R}^N$ at location $\mathbf{x}_i$ defined by

$$y_{i,d} = f_d(\mathbf{x}_i) + \epsilon_d, \tag{2}$$

where $\epsilon_d \sim \mathcal{N}(0, \sigma_d^2)$ and $\mathbf{f}_d \equiv [f_d(\mathbf{x}_1), \ldots, f_d(\mathbf{x}_N)]$ is function values with GP prior. This leads to the following conditional distribution:

$$p(\mathbf{y}_{:,d} \mid \mathbf{X}) = \mathcal{N}(\mathbf{y}_{:,d} \mid \mathbf{0}, \mathbf{K}_{NN} + \sigma_d^2 \mathbb{I}_N), \tag{3}$$

with $\mathbf{K}_{NN}$ being an $N \times N$ covariance matrix defined using a kernel function as a function of the latent space $\mathbf{X}$.

The ARD kernel is used for the kernel [29]. This kernel is generally used for determining the relevance of dimensions in $\mathbf{X}$ by tuning α_q. A typical ARD kernel is defined by:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{q=1}^{Q} \alpha_q (x_q - x_q')^2\right), \tag{4}$$

where $\mathbf{x}$ and $\mathbf{x}'$ represent two input vectors in the latent space. The term σ_f^2 denotes the variance parameter of the kernel, dictating the scale of the output space. Each α_q is an inverse length-scale parameter associated with the qth dimension of the latent space, which influences how variations in that particular dimension affect the kernel computation. These length-scale parameters allow the kernel to adjust the relevance of the feature dimensions.

The GPLVM can be interpreted as a variant of a multi-output Gaussian process. Every dimension of the observed data, $\mathbf{Y}$, is a separate realization of a Gaussian process, all derived by a common latent structure [30]. To fit GPLVM to a dataset, a suitable representation of the latent variables that best explain the observed data must be found. This can be done by estimating ARD kernel-free parameters. Three primary methods used for finding suitable representation include point estimate, Maximum A Posteriori (MAP) [15], and a fully Bayesian approach [31]. Whereas the point estimate method can be more susceptible to overfitting, the Bayesian approach typically offers a better generalization to data due to its integration over uncertainties. The payoff is that the Bayesian approach is computationally more expensive and often requires sophisticated inference techniques such as Markov Chain Monte Carlo (MCMC) [32] or variational inferencing [33]. This research focuses on Bayesian methods to derive a more robust inference. In the Bayesian framework, it is imperative to specify a prior distribution for the latent variables. It is assumed that the prior on the latent variables is Gaussian, defined by:

$$p(\mathbf{X}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{x}_n \mid \mathbf{0}, \mathbb{I}_Q), \tag{5}$$

where $\mathbf{x}_n$ denotes the nth row of the latent variables matrix $\mathbf{X}$. With this prior, the marginal probability of $\mathbf{Y}$ involves the integral, which is defined by:

$$\begin{aligned} p(\mathbf{Y}) &= \int p(\mathbf{Y} \mid \mathbf{X}) p(\mathbf{X}) d\mathbf{X} \\ &= \int \prod_{d=1}^{D} p(\mathbf{y}_{:,d} \mid \mathbf{X}) p(\mathbf{X}) d\mathbf{X}. \end{aligned} \tag{6}$$

This integral corresponds to the expectation of conditional probability of $\mathbf{Y}$ over $\mathbf{X}$. The integral does not have a closed-form solution, as $\mathbf{K}_{NN}$ is a function of $\mathbf{X}$. To find the marginal distribution, Jensen's inequality is applied, which provides a variational lower bound on the log likelihood:

$$\begin{aligned} \log p(\mathbf{Y}) \geq & \sum_{d=1}^{D} \mathbb{E}_{q(\mathbf{X})}[\log p(\mathbf{y}_{:,d} \mid \mathbf{X})] \\ & - \mathrm{KL}(q(\mathbf{X}) \parallel p(\mathbf{X} \mid \mathbf{Y})). \end{aligned} \tag{7}$$

It is necessary to pick a proper $\mathbf{q}$ distribution to achieve a tight bound here. The number of samples in the training set introduces additional complexities to the model; thus, we will use the inducing points to derive a more tractable solution to calculate the bound. Different strategies have been proposed to calculate the bound, including the solution proposed by [19], which creates a closed-form solution for the bound. [34] employed the variational distribution to simplify the problem by enabling lower-dimensional sampling conducive to convergence. Here, we discuss our approach, which integrates the labels or categories attached to each data point into the formation of our low-dimensional representation, thereby facilitating the classification of these labels. This has been accomplished through the doubly variational method, upon which the LDGD has been developed.

B. LATENT DISCRIMINATIVE GENERATIVE DECODER MODEL

While the idea behind LDGD is the same as GPVLM, our objective is different. In LDGD, the aim is to infer the latent process that can capture essential features of the observed data and separate the classes in the data, contrasting with GPLVM, which primarily focuses on dimensionality reduction without explicitly optimizing for class separability. Essentially, the goal is to identify a set of features in the data that can be used for both data generation and achieving high accuracy in the data class or category prediction. It is noted that LDGD inference will differ from those derived by generative auto-encoders, as the inferred latent process needs to capture those essential features of the data that are expressive of labels or categories. The following section will delve into the framework definition and its different modeling steps, including training and prediction.

1) OBSERVED DATA

In LDGD, It is assumed the observed data consists of continuous measures, represented as $\mathbf{Y}^r \in \mathbb{R}^{N\times D}$, and corresponding classes of data or categories, represented as $\mathbf{Y}^c \in \mathbb{R}^{N\times K}$. N denotes the number of samples, D is the dimension of continuous measures per sample, and K is the number of distinct categories or classes per sample. Each row of $\mathbf{Y}^r$, denoted as $\mathbf{y}^r_i$, corresponds to the continuous measurement for the ith sample. Each column, denoted as $\mathbf{y}^r_{:,d}$, represents the data for the dth dimension across all samples. Similarly, each row of $\mathbf{Y}^c$, denoted as $\mathbf{y}^c_i$, corresponds to the one-hot encoded class label for each sample.

2) LATENT VARIABLES

LDGD incorporates latent variables $\mathbf{X} \in \mathbb{R}^{N\times Q}$, to capture the essential features driving both $\mathbf{Y}^r$ and $\mathbf{Y}^c$. Here, Q represents the latent space's dimensionality, which is generally much smaller than the dimension of the observed data. A Gaussian prior is assumed for these latent variables, expressed as:

$$p(\mathbf{X}) = \prod_{i=1}^{N} \mathcal{N}(\mathbf{x}_i \mid \mathbf{0}, \mathbb{I}_Q), \tag{8}$$

where $\mathbf{0}$ is a zero vector and $\mathbb{I}_Q$ is the identity matrix of size $Q \times Q$.

3) GAUSSIAN PROCESS PRIORS

To model the relationships between the latent space and the observed data, two Gaussian process (GP) priors are imposed on function values $\mathbf{F}^r$ and $\mathbf{F}^c$. Specifically, The GP priors are defined over $\mathbf{F}^r \in \mathbb{R}^{N\times D}$, for the continuous measurements, and $\mathbf{F}^c \in \mathbb{R}^{N\times K}$ for the categorical measurements or labels. These prior distributions are defined by:

$$p(\mathbf{F}^c \mid \mathbf{X}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{f}^c_{:,k} \mid \mathbf{0}, \mathbf{K}^c_{NN}) \tag{9}$$

$$p(\mathbf{F}^r \mid \mathbf{X}) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{f}^r_{:,d} \mid \mathbf{0}, \mathbf{K}^r_{NN}), \tag{10}$$

where $\mathbf{K}^c_{NN}$ and $\mathbf{K}^r_{NN}$ are the $N \times N$ covariance matrices for the GPs corresponding to class labels and observed measurements, respectively. The element in the ith row and jth column of these matrices are defined by kernel functions $k^c_{i,j} = k_\psi(\mathbf{x}_i, \mathbf{x}_j)$ and $k^r_{i,j} = k_\theta(\mathbf{x}_i, \mathbf{x}_j)$, with θ and ψ as the kernel parameters. Furthermore, $\mathbf{f}^c_{:,k}$ represents the kth column of $\mathbf{F}^c$, and $\mathbf{f}^r_{:,d}$ denotes the dth column of $\mathbf{F}^r$. The independence assumption embedded in Equation (9) and Equation (10) is a direct consequence of the dual formulation in PPCA. A detailed explanation of duality is provided in the Appendix B.

4) OBSERVED DATA MODEL

LDGD delineates the relationship between the features and the observed data. This relationship is modeled by:

$$p(\mathbf{Y}^r \mid \mathbf{F}^r) = \prod_{i=1}^{N}\prod_{d=1}^{D} \mathcal{N}(y^r_{i,d} \mid f^r_d(\mathbf{x}_i), \sigma^2_d) \tag{11}$$

$$p(\mathbf{Y}^c \mid \mathbf{F}^c) = \prod_{i=1}^{N}\prod_{k=1}^{K} \text{Bernoulli}(y^c_{i,k} \mid g(f^c_k(\mathbf{x}_i))), \tag{12}$$

where σ^2_d is the variance of the Gaussian noise and $g(\cdot)$ is a squash function, such as the Sigmoid or Probit, ensuring the output represents a probability. For the ith element in the dth column of $\mathbf{F}^r$, $f^r_{i,d} \triangleq f^r_d(\mathbf{x}_i)$ is defined. Similarly, for the ith element in the kth column of $\mathbf{F}^c$, $f^c_{i,k} \triangleq f^c_k(\mathbf{x}_i)$ is defined. Figure 1 shows the graphical model behind the LDGD framework.

5) MARGINAL LIKELIHOOD LOWER BOUND

Equations (8) to (12) define the LDGD model. It is necessary to find the marginal distribution of Y^r and Y^c to fit the model to the dataset. The marginal distribution is defined by:

$$p(\mathbf{Y}^r, \mathbf{Y}^c) = \int p(\mathbf{Y}^r, \mathbf{Y}^c, \mathbf{F}^r, \mathbf{F}^c, \mathbf{X})\, d\mathbf{F}^r\, d\mathbf{F}^c\, d\mathbf{X}. \tag{13}$$

Similar to Equation (6), this integral has no closed-form solution. Additionally, calculating the integral numerically is impractical due to the high dimensionality of the latent elements over which the integral is taken. Given there is no solution to find this integral, maximizing the evidence by adjusting its free parameter becomes impossible. Instead, a lower bound for the marginal likelihood is derived, similar to what is shown in Equation (64). To achieve this, inducing points and variational distributions are used. As shown later, this solution will have a modest computational cost and can be applied to high-dimensional datasets. Besides, model training can be done through batches of datasets, making it appropriate for iterative training.

a: INDUCING POINTS AND VARIATIONAL DISTRIBUTIONS

In Appendix A, the utilization of inducing points for GP regression is explained. Similarly, in LDGD, the same concept is applied, with the distinction that both discrete and continuous observation processes need to be addressed. Attributes of these processes impose different sensitivities to the choice of inducing points; therefore, two sets of inducing points are selected, one for the discrete and one for the continuous observations.

For the label data in the classification component, the inducing points and variables are defined as

$$p(\mathbf{U}^c \mid \mathbf{Z}^c) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{u}^c_{:,k} \mid \mathbf{0}, k^c(\mathbf{Z}^c, \mathbf{Z}^c)), \tag{14}$$

where $\mathbf{U}^c \equiv \left[\mathbf{u}^c_{:,1}, \ldots, \mathbf{u}^c_{:,K}\right]$, $\mathbf{u}^c_{:,k} \in \mathbb{R}^M$, and $\mathbf{K}^c_{M_cM_c} \triangleq k^c(\mathbf{Z}^c, \mathbf{Z}^c)$. Here, $\mathbf{Z}^c$ represents the set with M_c inducing points associated with the $\mathbf{F}^c$. Analogously, for the continuous measurement of the regression component, the following is obtained:

$$p(\mathbf{U}^r \mid \mathbf{Z}^r) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{u}^r_{:,d} \mid \mathbf{0}, k^r(\mathbf{Z}^r, \mathbf{Z}^r)), \tag{15}$$

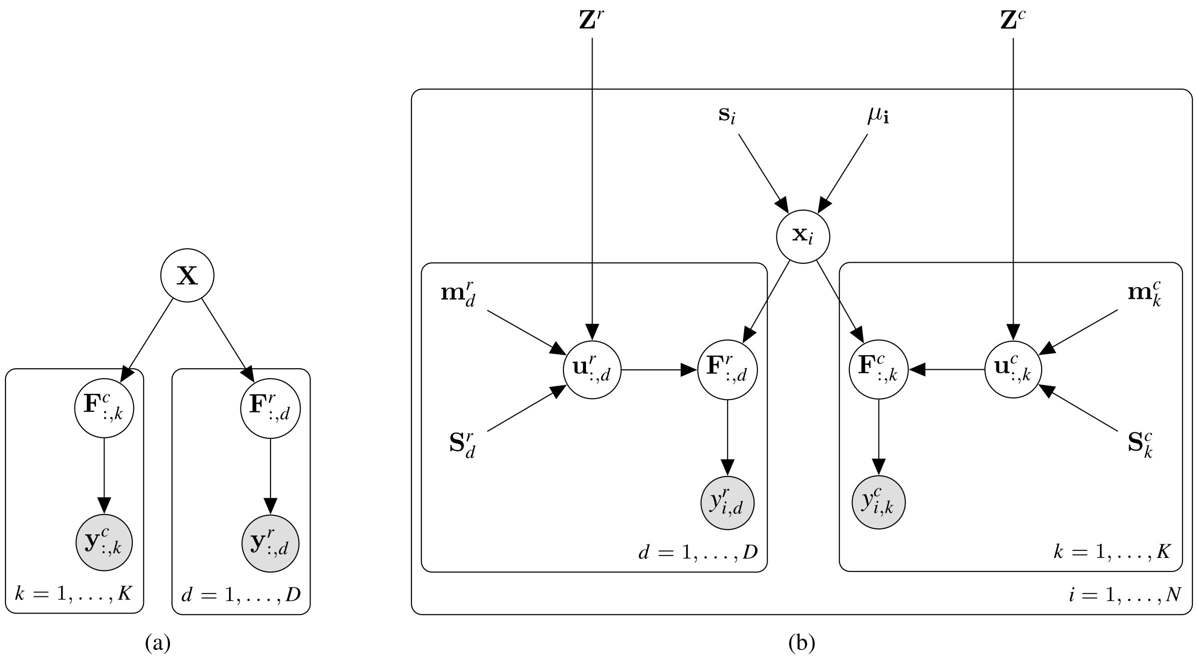

FIGURE 1. Graphical models depicting LDGD. (a) Exact inference, (b) Variational inference.

where $\mathbf{U}^r \equiv \left[\mathbf{u}^r_{:,1},\dots,\mathbf{u}^r_{:,D}\right]$, $\mathbf{u}^r_{:,d} \in \mathbb{R}^M$, and $\mathbf{K}^r_{M_rM_r} \triangleq k^r(\mathbf{Z}^r,\mathbf{Z}^r)$. Here, a new set of M_r inducing points, Z^r, associated with $\mathbf{F}^r$ is used. The choice of two disjoint sets of inducing points addresses the different sensitivities in the regression and classification processes.

b: DOUBLY STOCHASTIC VARIATIONAL GP AND VARIATIONAL POSTERIOR DISTRIBUTION

The true posterior distribution is given by

$$p(\mathbf{F}^r,\mathbf{F}^c,\mathbf{X}\mid\mathbf{Y}^r,\mathbf{Y}^c)=\frac{p(\mathbf{F}^r,\mathbf{F}^c,\mathbf{X},\mathbf{Y}^r,\mathbf{Y}^c)}{p(\mathbf{Y}^r,\mathbf{Y}^c)}. \tag{16}$$

Calculating the denominator involves the computation of Equation (13), which is intractable. We propose variational distributions as a part of our approach to approximate the posterior distribution in LDGD. Our methodology incorporates concepts from the doubly stochastic variational Gaussian process (DSVGP) outlined by [34], and integrates the scalable variational GP framework for classification proposed by [35]. This integration is achieved through the utilization of two distinct sets of inducing variables, as delineated in Equations (14) and (15), which are applied respectively for classification and regression tasks. Training involves maximizing the marginal likelihood (evidence), and our method provides a mean to find a lower bound for the otherwise intractable marginalization process. To achieve this lower bound, the process begins by marginalizing the joint distribution of LDGD, taking into account the variational inducing points introduced:

$$p(\mathbf{Y}^r,\mathbf{Y}^c)=\int p(\mathbf{Y}^r,\mathbf{Y}^c,\mathbf{F}^r,\mathbf{F}^c,\mathbf{U}^r,\mathbf{U}^c,\mathbf{X})\times d\mathbf{F}^r\,d\mathbf{F}^c\,d\mathbf{U}^r\,d\mathbf{U}^c\,d\mathbf{X}. \tag{17}$$

The joint distribution of LDGD, including the observed data $\mathbf{Y}^r$ and $\mathbf{Y}^c$, the function values $\mathbf{F}^r$ and $\mathbf{F}^c$, the inducing variables $\mathbf{U}^r$ and $\mathbf{U}^c$, and the latent features $\mathbf{X}$, can be factorized as:

$$\begin{aligned}p(\mathbf{F}^r,\mathbf{F}^c,\mathbf{U}^r,\mathbf{U}^c,\mathbf{X},\mathbf{Y}^r,\mathbf{Y}^c)&=p(\mathbf{Y}^c\mid\mathbf{F}^c)p(\mathbf{F}^c,\mathbf{U}^c\mid\mathbf{X})\\&\quad\times p(\mathbf{Y}^r\mid\mathbf{F}^r)p(\mathbf{F}^r,\mathbf{U}^r\mid\mathbf{X})\\&\quad\times p(\mathbf{X}).\end{aligned} \tag{18}$$

Solving marginalization integral is not straightforward. A common technique used in variational inference is multiplying and dividing the terms inside the integral by a proposal distribution, which is an approximation of the joint posterior. The joint posterior $p(\mathbf{F}^r,\mathbf{F}^c,\mathbf{U}^r,\mathbf{U}^c,\mathbf{X}\mid\mathbf{Y}^r,\mathbf{Y}^c)$ can be factorized into three separate posteriors:

$$\begin{aligned}p(\mathbf{F}^r,\mathbf{F}^c,\mathbf{U}^r,\mathbf{U}^c,\mathbf{X}\mid\mathbf{Y}^r,\mathbf{Y}^c)&=p(\mathbf{F}^c,\mathbf{U}^c\mid\mathbf{Y}^c,\mathbf{X})\\&\quad\times p(\mathbf{F}^r,\mathbf{U}^r\mid\mathbf{Y}^r,\mathbf{X})\\&\quad\times p(\mathbf{X}\mid\mathbf{Y}^r,\mathbf{Y}^c).\end{aligned} \tag{19}$$

where $p(\mathbf{F}^c,\mathbf{U}^c\mid\mathbf{Y}^c,\mathbf{X})$ and $p(\mathbf{F}^r,\mathbf{U}^r\mid\mathbf{Y}^r,\mathbf{X})$ are the joint posterior of the Gaussian process function values and their inducing variables for regression and classification respectively. $p(\mathbf{X}\mid\mathbf{Y}^r,\mathbf{Y}^c)$ shows the posterior over the latent features. To derive a tractable solution using the posterior, the posterior $p(\mathbf{X}\mid\mathbf{Y}^r,\mathbf{Y}^c)$ is approximated with the variational distribution $q_\phi(\mathbf{X})$,

$$q_\phi(\mathbf{X})=\prod_{i=1}^{N}\mathcal{N}(\mathbf{x}_i\mid\boldsymbol{\mu}_i,\mathbf{s}_i\mathbb{I}_Q), \tag{20}$$

Also $p(\mathbf{F}^c,\mathbf{U}^c\mid\mathbf{Y}^c,\mathbf{X})$ is approximated with $q(\mathbf{F}^c,\mathbf{U}^c)$

$$q(\mathbf{F}^c,\mathbf{U}^c)=p(\mathbf{F}^c\mid\mathbf{U}^c,\mathbf{X})q_\lambda(\mathbf{U}^c)$$

$$q_\lambda(\mathbf{U}^c) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{u}_k^c \mid \mathbf{m}_k^c, \mathbf{S}_k^c), \tag{21}$$

and $p(\mathbf{F}^r, \mathbf{U}^r \mid \mathbf{Y}^r, \mathbf{X})$ is approximated with $q(\mathbf{F}^r, \mathbf{U}^r)$:

$$\begin{aligned} q(\mathbf{F}^r, \mathbf{U}^r) &= p(\mathbf{F}^r \mid \mathbf{U}^r, \mathbf{X}) q_\gamma(\mathbf{U}^r), \\ q_\gamma(\mathbf{U}^r) &= \prod_{d=1}^{D} \mathcal{N}(\mathbf{u}_d^r \mid \mathbf{m}_d^r, \mathbf{S}_d^r), \end{aligned} \tag{22}$$

where $\phi = \{\boldsymbol{\mu}_i, \mathbf{s}_i\}_{i=1}^N$, $\lambda = \{\mathbf{m}_k^c, \mathbf{S}_k^c\}_{k=1}^K$,and $\gamma = \{\mathbf{m}_d^r, \mathbf{S}_d^r\}_{d=1}^D$ are the variational parameters. The joint posteriors $p(\mathbf{F}^r, \mathbf{F}^c, \mathbf{U}^r, \mathbf{U}^c, \mathbf{X} \mid \mathbf{Y}^r, \mathbf{Y}^c)$ can be approximated using the variational distribution $q(\mathbf{F}^r, \mathbf{F}^c, \mathbf{U}^r, \mathbf{U}^c, \mathbf{X})$ that can be expressed as:

$$\begin{aligned} \hat{Q} &\triangleq q(\mathbf{F}^r, \mathbf{F}^c, \mathbf{U}^r, \mathbf{U}^c, \mathbf{X}) \\ &\equiv q(\mathbf{F}^c, \mathbf{U}^c) q(\mathbf{F}^r, \mathbf{U}^r) q_\phi(\mathbf{X}). \end{aligned} \tag{23}$$

Employing the approximated posterior, the evidence introduced in Equation (17) is reformulated, and Jensen's inequality is applied to the logarithm of the evidence to establish the evidence lower-bound (ELBO):

$$\log p(\mathbf{Y}^r, \mathbf{Y}^c) \geq E_{\hat{Q}}\left[\log P\right] \triangleq \text{ELBO}, \tag{24}$$

where

$$P = \frac{p(\mathbf{Y}^c \mid \mathbf{F}^c) p(\mathbf{U}^c) p(\mathbf{Y}^r \mid \mathbf{F}^r) p(\mathbf{U}^r) p(\mathbf{X})}{q_\phi(\mathbf{X}) q_\lambda(\mathbf{U}^c) q_\gamma(\mathbf{U}^r)}. \tag{25}$$

This ELBO can be further broken down into five terms:

$$\text{ELBO} \equiv \text{ELL}^{\text{reg}} + \text{ELL}^{\text{cls}} - \text{KL}_u^c - \text{KL}_u^r - \text{KL}_X, \tag{26}$$

where the ELL^{reg} and ELL^{cls} are expected log-likelihood (ELL) for regression and classification paths, respectively, and are defined by:

$$\begin{aligned} \text{ELL}^{\text{reg}} &\equiv E_{q_\phi(\mathbf{X})}\left[E_{q(\mathbf{F}^r, \mathbf{U}^r)}\left[\log p(\mathbf{Y}^r \mid \mathbf{F}^r)\right]\right], \quad (27) \\ \text{ELL}^{\text{cls}} &\equiv E_{q_\phi(\mathbf{X})}\left[E_{q(\mathbf{F}^c, \mathbf{U}^c)}\left[\log p(\mathbf{Y}^c \mid \mathbf{F}^c)\right]\right]. \quad (28) \end{aligned}$$

The KL terms are defined as:

$$\begin{aligned} \text{KL}_u^c &\equiv KL(q_\lambda(\mathbf{U}^c) \parallel p(\mathbf{U}^c)), \quad (29) \\ \text{KL}_u^r &\equiv KL(q_\gamma(\mathbf{U}^r) \parallel p(\mathbf{U}^r)), \quad (30) \\ \text{KL}_X &\equiv KL(q_\phi(\mathbf{X}) \parallel p(\mathbf{X})). \quad (31) \end{aligned}$$

ELL^{reg} and ELL^{cls} contribute to data fitting, whereas KL_u^c, KL_u^r, and KL_X terms serve as regularizers and help prevent overfitting. Minimizing the KL divergence between the posterior defined in Equation (19) and its variational estimation defined in Equation (23) is equivalent to maximizing the ELBO. A detailed explanation is provided in Appendix C. A closed form for this lower bound is desirable for optimizing the variational parameters, inducing points, and kernel parameters. While KL terms have a closed form due to the Gaussian distribution characteristics, ELL^{cls} and ELL^{reg} do not have a closed form in general. The details of calculating these two terms are explained in the subsequent sections.

6) CLASSIFICATION EXPECTED LOG-LIKELIHOOD (ELL^{CLS})

In this section, a detailed analysis of ELL^{cls} is provided, defined as:

$$\begin{aligned} \text{ELL}^{\text{cls}} &= E_{q_\phi(\mathbf{X})}\left[E_{p(\mathbf{F}^c \mid \mathbf{U}^c, \mathbf{X}) q_\lambda(\mathbf{U}^c)}\left[\log p(\mathbf{Y}^c \mid \mathbf{F}^c)\right]\right] \\ &= \sum_{i,k} ELL_{i,k}^{\text{cls}}. \end{aligned} \tag{32}$$

The $\text{ELL}_{i,k}^{\text{cls}}$ is rewritten as:

$$\begin{aligned} ELL_{i,k}^{\text{cls}} &= \int q_\phi(\mathbf{x}_i) p(f_k^c \mid \mathbf{u}_k^c, \mathbf{x}_i) q_\lambda(\mathbf{u}_k^c) \log p(y_{i,k}^c \mid f_k^c(\mathbf{x}_i)) \\ &\quad \times df_k^c \, d\mathbf{u}_k^c \, d\mathbf{x}_i \\ &= \int q_\phi(\mathbf{x}_i) q_\lambda(f_k^c \mid \mathbf{x}_i) \log p(y_{i,k}^c \mid f_k^c(\mathbf{x}_i)) \, df_k^c \, d\mathbf{x}_i \\ &= E_{q_\phi(\mathbf{x}_i) q_\lambda(f_k^c \mid \mathbf{x}_i)}\left[\log p(y_{i,k}^c \mid f_k^c(\mathbf{x}_i))\right], \end{aligned} \tag{33}$$

where the predictive distribution is defined as:

$$\begin{aligned} q_\lambda(f_k^c \mid \mathbf{x}_i) &\triangleq \int p(f_k^c \mid u_k^c, \mathbf{x}_i) q_\lambda(\mathbf{u}_k^c) \, d\mathbf{u}_k^c \\ &= \mathcal{N}\left(\boldsymbol{\mu}_{f_k^c}(\mathbf{x}_i), \boldsymbol{\Sigma}_{f_k^c}(\mathbf{x}_i)\right). \end{aligned} \tag{34}$$

By defining $\Psi_c = \mathbf{k}_{iM_c}^c (\mathbf{K}_{M_c M_c}^c)^{-1}$, the mean vector and covariance matrix of the predictive distribution can be expressed as:

$$\begin{aligned} \boldsymbol{\mu}_{f_k^c}(\mathbf{x}_i) &= \Psi_c \mathbf{m}_k^c, \quad (35) \\ \boldsymbol{\Sigma}_{f_k^c}(\mathbf{x}_i) &= k_{ii}^c - \Psi_c(\mathbf{S}_d^c - \mathbf{K}_{M_c M_c}^c)\Psi_c^T, \quad (36) \end{aligned}$$

with $\mathbf{k}_{iM_c}^c = k^c(\mathbf{x}_i, \mathbf{Z}^c)$ being the ith row of $\mathbf{K}_{NM_c}^c$, and $k_{ii}^c = k^c(\mathbf{x}_i, \mathbf{x}_i)$. The details are provided in Appendix D. For ease of use, we define $B(\mathbf{x}_i) \triangleq E_{q_\gamma(\mathbf{f}_k^c \mid \mathbf{x}_i)}\left[\log p(y_{i,k}^c \mid f_k^c(\mathbf{x}_i))\right]$. For the Bernoulli likelihood with Probit likelihood function, the inner expectation $B(\mathbf{x}_i)$ is:

$$\begin{aligned} B(\mathbf{x}_i) &= E_{q_\gamma(\mathbf{f}_k^c \mid \mathbf{x}_i)}\left[\log p(y_{i,k}^c \mid f_k^c(\mathbf{x}_i))\right] \\ &= \int q_\gamma(\mathbf{f}_k^c \mid \mathbf{x}_i) \log \Phi((2y_{i,k}^c - 1) f_k^c(\mathbf{x}_i)) \, df_k^c \\ &= \int \frac{1}{\sqrt{2\pi \, \boldsymbol{\Sigma}_{f_k^c}(\mathbf{x}_i)}} \exp\left(-\frac{(\mathbf{f}_k^c - \boldsymbol{\mu}_{f_k^c}(\mathbf{x}_i))^2}{2\boldsymbol{\Sigma}_{f_k^c}(\mathbf{x}_i)}\right) \\ &\quad \times \log \Phi((2y_{i,k}^c - 1) f_k^c(\mathbf{x}_i)) \, d\mathbf{f}_k^c. \end{aligned} \tag{37}$$

To use Gauss-Hermite quadrature, this integral needs to be transformed into the standard form of $\int_{-\infty}^{\infty} e^{-x^2} g(x) \, dx$. A change of variable is performed: $\mathbf{z} = \frac{\mathbf{f}_k^c - \boldsymbol{\mu}_{f_k^c}(\mathbf{x}_i)}{\sqrt{2\boldsymbol{\Sigma}_{\mathbf{f}_k^c}(\mathbf{x}_i)}}$ so that $\mathbf{f}_k^c = \mathbf{z}\sqrt{2\boldsymbol{\Sigma}_{\mathbf{f}_k^c}(\mathbf{x}_i)} + \boldsymbol{\mu}_{f_k^c}(\mathbf{x}_i)$ and $d\mathbf{f}_k^c = \sqrt{2\boldsymbol{\Sigma}_{f_k^c}(\mathbf{x}_i)} \, dz$. The integral then becomes:

$$B(\mathbf{x}_i) = \int_{-\infty}^{\infty} \frac{e^{-z^2}}{\sqrt{\pi}} g(z) \, dz. \tag{38}$$

Now, the integral is in a form suitable for Gauss-Hermite quadrature, where

$$g(z) = \sqrt{\boldsymbol{\Sigma}_{\mathbf{f}_k^c}(\mathbf{x}_i)} \log \Phi((2y_{i,k}^c - 1)(\mathbf{z}\sqrt{2\boldsymbol{\Sigma}_{f_k^c}(\mathbf{x}_i)} + \boldsymbol{\mu}_{f_k^c}(\mathbf{x}_i))). \tag{39}$$

Let $z_i^{(j)}$ and $w^{(j)}$ denote the nodes and weights of the Gauss-Hermite quadrature. The integral $B(\mathbf{x}_i)$ can be approximated as:

$$B(\mathbf{x}_i) \approx \frac{1}{\sqrt{\pi}} \sum_{l=1}^{L} w^{(l)} g(z_i^{(l)}), \tag{40}$$

where L is the number of quadrature points. The $z_i^{(l)}$ are the roots of the physicists' version of the Hermite polynomial $H_n(z_i)$, and the associated weights $w^{(l)}$ are given by

$$w^{(l)} = \frac{2^{L-1} L! \sqrt{\pi}}{L^2 [H_{L-1}(z_i)]^2}. \tag{41}$$

This approximation allows for the numerical evaluation of the integral using the Gauss-Hermite quadrature method [36].To calculate $E_{q_\phi(\mathbf{x}_i)}[B(\mathbf{x}_i)]$, J samples can be drawn from $\mathbf{x}_i^{(j)} \sim q_\phi(\mathbf{x}_i)$ to estimate the expectation.

$$E_{q_\phi(\mathbf{x}_i)}[B(\mathbf{x}_i)] \approx \frac{1}{\sqrt{\pi}} \sum_{l=1}^{L} \sum_{j=1}^{J} w^{(l)} g(z_i^{(l)}). \tag{42}$$

7) REGRESSION EXPECTED LOG LIKELIHOOD (ELL^{reg})

In this section, the regression component of the model is addressed. With a similar calculation, the following is obtained:

$$\begin{aligned} \mathrm{ELL}^{\mathrm{reg}} &= E_{q_\phi(\mathbf{X})}\left[E_{p(\mathbf{F}^r|\mathbf{U}^r,\mathbf{X})q_\lambda(\mathbf{U}^r)}\left[\log p(\mathbf{Y}^r \mid \mathbf{F}^r)\right]\right] \\ &= \sum_{i,d} ELL_{i,d}^{\mathrm{reg}}. \end{aligned} \tag{43}$$

Here are the computation details for calculating the ELL:

$$\begin{aligned} ELL_{i,d}^{\mathrm{reg}} &= \int q_\phi(\mathbf{x}_i) p(\mathbf{f}_d^r \mid \mathbf{u}_d^r, \mathbf{x}_i) q_\gamma(\mathbf{u}_d^r) \\ &\quad \times \log p(y_{i,d}^r \mid f_d^r(\mathbf{x}_i))\, d\mathbf{f}_d^r\, d\mathbf{u}_d^r\, d\mathbf{x}_i \\ &= \int q_\phi(\mathbf{x}_i) q_\gamma(\mathbf{f}_d^r \mid \mathbf{x}_i) \log p(y_{i,d}^r \mid f_d^r(\mathbf{x}_i))\, d\mathbf{f}_d^r\, d\mathbf{x}_i \\ &= E_{q_\phi(\mathbf{x}_i)}\left[E_{q_\gamma(\mathbf{f}_d^r|\mathbf{x}_i)}\left[\log p(y_{i,d}^r \mid f_d^r(\mathbf{x}_i))\right]\right] \\ &= E_{q_\phi(\mathbf{x}_i)}[A(\mathbf{x}_i)]. \end{aligned} \tag{44}$$

It is already known that $y_{i,d}^r = f_d^r(\mathbf{x}_i) + \epsilon_d$, therefore the internal term has a Gaussian distribution:

$$p(y_{i,d}^r \mid f_d^r(\mathbf{x}_i)) = \mathcal{N}(f_d^r(\mathbf{x}_i), \boldsymbol{\Sigma}_d^2). \tag{45}$$

By defining $\Psi_r = \mathbf{k}_{iM_r}^r (\mathbf{K}_{M_rM_r}^r)^{-1}$, a closed form for the predictive distribution can be derived:

$$q_\gamma(f_d^r \mid \mathbf{x}_i) = \mathcal{N}\left(f_d^r \mid \boldsymbol{\mu}_{\mathbf{f}_d^r}(\mathbf{x}_i), \boldsymbol{\Sigma}_{\mathbf{f}_d^r}(\mathbf{x}_i)\right) \tag{46}$$

$$\boldsymbol{\mu}_{\mathbf{f}_d^r}(\mathbf{x}_i) = \Psi_r \mathbf{m}_d^r \tag{47}$$

$$\boldsymbol{\Sigma}_{\mathbf{f}_d^r}(\mathbf{x}_i) = k_{ii}^r + \Psi_r(\mathbf{S}_d^r - \mathbf{K}_{M_rM_r}^r)\Psi_r^T, \tag{48}$$

where $\mathbf{k}_{iM_r}^r = k^r(\mathbf{x}_i, \mathbf{Z}^r)$, $k_{ii}^r = k^r(\mathbf{x}_i, \mathbf{x}_i)$, and $K_{M_rM_r}^r$ is the covariance matrix evaluated at inducing points.

$$\begin{aligned} A(\mathbf{x}_i) &= E_{q_\gamma(f_d^r|\mathbf{x}_i)}\left[\log p(y_{i,d}^r \mid f_d^r(\mathbf{x}_i))\right] \\ &= \int q_\gamma(f_d^r \mid \mathbf{x}_i) \log p(y_{i,d}^r \mid f_d^r(\mathbf{x}_i))\, df_d^r \\ &= -0.5 \left[\begin{array}{l} \log 2\pi + \log \boldsymbol{\Sigma}_d^2 + \\ \dfrac{(y_{i,d}^r - \boldsymbol{\mu}_{\mathbf{f}_d^r}(\mathbf{x}_i))^2}{\boldsymbol{\Sigma}_d^2} + \dfrac{\boldsymbol{\Sigma}_{\mathbf{f}_d^r}(\mathbf{x}_i)}{\boldsymbol{\Sigma}_d^2} \end{array}\right]. \end{aligned} \tag{49}$$

This can be easily calculated by sampling.

$$\begin{aligned} ELL_{i,d}^{\mathrm{reg}} &= E_{q_\phi(\mathbf{x}_i)}\left[E_{q_\gamma(f_d^r|\mathbf{x}_i)}\left[\log p(y_{i,d}^r \mid f_d^r(\mathbf{x}_i))\right]\right] \\ &= -\frac{1}{2J} \sum_{j=1}^{J} \left[\begin{array}{l} \log 2\pi + \log \boldsymbol{\Sigma}_d^2 + \\ \dfrac{(y_{i,d}^r - \boldsymbol{\mu}_{\mathbf{f}_d^r}(x_i^{(j)}))^2 + \boldsymbol{\Sigma}_{\mathbf{f}_d^r}(x_i^{(j)})}{\boldsymbol{\Sigma}_d^2} \end{array}\right]. \end{aligned} \tag{50}$$

It is worth mentioning that for some certain kernel functions, finding the analytic form for the expectation introduced in Equation (44) is possible.

8) TRAINING PROCEDURE

Learning occurs through optimizing the kernel hyperparameters, inducing points, and variational parameters by maximizing the ELBO as the objective function.

$$\underset{\theta,\phi,\lambda,\gamma,\mathbf{Z}^r,\mathbf{Z}^c}{\arg\max} \quad \mathrm{ELBO} \tag{51}$$

where θ represents the kernel hyperparameters, $\phi = \{(\boldsymbol{\mu}_i, \mathbf{s}_i), i = 1, \ldots, N\}$ are the variational parameters for latent features, $\lambda = \{(\mathbf{m}_k^c, \mathbf{S}_k^c), k = 1, \ldots, K\}$ and $\gamma = \{(\mathbf{m}_d^r, \mathbf{S}_d^r), d = 1, \ldots, D\}$ are the variational parameters for inducing variables. In the previous section, it was discussed how the expected log-likelihood for regression (ELL^{reg}) and classification (ELL^{cls}) can be broken down and calculated for each sample in the dataset. This formulation simplifies the calculations and enables batch training and sampling methods. To optimize the variational parameters for $q_\phi(\mathbf{X})$ during sampling, samples $\mathbf{x}_i^{(j)} \sim q_\phi(\mathbf{x}_i)$ are generated using the reparameterization technique [37], where $\epsilon^{(j)} \sim \mathcal{N}(0, \mathbb{I}_Q)$ and $\mathbf{x}_i^{(j)} = \boldsymbol{\mu}_i + \mathbf{s}_i \odot \epsilon^{(j)}$. Based on Equations (26), (44), and (32), the training procedure is simplified to this maximization problem:

$$\begin{aligned} \underset{\theta,\phi,\lambda,\gamma,\mathbf{Z}^r,\mathbf{Z}^c}{\arg\max} & \sum_{i,d} \mathrm{ELL}_{i,d}^{\mathrm{reg}} + \sum_{i,k} \mathrm{ELL}_{i,k}^{\mathrm{cls}} \\ & - \sum_{i=1}^{N} \mathrm{KL}(q_\phi(\mathbf{x}_i) \parallel p(\mathbf{x}_i)) \\ & - \sum_{k=1}^{K} \mathrm{KL}(q_\lambda(\mathbf{u}_k^c) \parallel p(\mathbf{u}_k^c \mid \mathbf{Z}^c)) \\ & - \sum_{d=1}^{D} \mathrm{KL}(q_\gamma(\mathbf{u}_d^r) \parallel p(\mathbf{u}_d^r \mid \mathbf{Z}^r)) \end{aligned} \tag{52}$$

where the KL divergence terms have closed forms due to the conjugacy of the Gaussian distributions.$\mathrm{ELL}_{i,d}^{\mathrm{reg}}$ and $\mathrm{ELL}_{i,k}^{\mathrm{cls}}$ was introduced in Equation (50) and (42). The Algorithm 1 details the training procedure. Detailed implementations of the predictive distribution, along with the calculations for the expected log-likelihood for regression and classification, are provided in Appendices D and E, respectively.

Algorithm 1 Training the Gaussian Process Regression Model

```
Require:
    Batch size ($B$)
    Learning rate ($lr$)
    Number of iterations ($N_{iter}$)
    Training continuous measures $\mathbf{Y}^r \in \mathbb{R}^{N\times D}$
    Training class labels $\mathbf{Y}^c \in \mathbb{R}^{N\times K}$
 1: procedure TrainModel($\mathbf{Y}^r, \mathbf{Y}^c, \mathbf{Z}^c, \mathbf{Z}^r, lr, N_{iter}$)
 2:     Initialize free parameters
            $\boldsymbol{\Theta} \triangleq \{\theta, \psi, \phi, \lambda, \gamma, \mathbf{Z}^r, \mathbf{Z}^c\}$
 3:     for $i = 1$ to $N_{iter}$ do
 4:         Sample $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_Q)$ for $i = 1, \dots, N$
 5:         Reparametrization $\mathbf{x}_i \leftarrow \boldsymbol{\mu}_i + \sqrt{\mathbf{s}_i}\boldsymbol{\epsilon}_i$
 6:         Sample a batch $\mathbf{X}_b$
 7:         Calculate $K^r_{NN}, \mathbf{K}^r_{M_rM_r}, \mathbf{K}^r_{M_rN}, K^c_{NN}, \mathbf{K}^c_{M_cM_c}, \mathbf{K}^c_{M_cN}$
 8:         $\boldsymbol{\mu}_{\mathbf{f}^r}, \boldsymbol{\Sigma}_{\mathbf{f}^r} \leftarrow PredictiveDist(\mathbf{X}_b, K^r_{NN}, \mathbf{K}^r_{M_rM_r}, \mathbf{K}^r_{M_rN}, \mathbf{m}^r, \mathbf{S}^r)$
 9:         $\boldsymbol{\mu}_{\mathbf{f}^c}, \boldsymbol{\Sigma}_{\mathbf{f}^c} \leftarrow PredictiveDist(\mathbf{X}_b, K^c_{NN}, \mathbf{K}^c_{M_cM_c}, \mathbf{K}^c_{M_cN}, \mathbf{m}^c, \mathbf{S}^c)$
10:         $\text{ELL}^{\text{reg}} \leftarrow ELLRegression(\mathbf{Y}^r, \boldsymbol{\mu}_{\mathbf{f}^r}, \boldsymbol{\Sigma}_{\mathbf{f}^r}, \boldsymbol{\sigma}^2, \mathbf{X}_b)$
11:         $\text{ELL}^{\text{cls}} \leftarrow \mathbf{ELLClassification}(\mathbf{Y}^c, \boldsymbol{\mu}_{\mathbf{f}^c}, \boldsymbol{\Sigma}_{\mathbf{f}^c}, \mathbf{X}_b)$
12:         $\text{KL}_U \leftarrow \text{KL}(q_\lambda(\mathbf{U}^c) \parallel p(\mathbf{U}^c \mid \mathbf{Z}^c)) + \text{KL}(q_\gamma(\mathbf{U}^r) \parallel p(\mathbf{U}^r \mid \mathbf{Z}^r))$
13:         $\text{ELBO} \leftarrow \text{ELL}^{\text{reg}} + \text{ELL}^{\text{cls}} - \text{KL}_U - \text{KL}(q_\phi(\mathbf{X}) \parallel p(\mathbf{X}))$
14:         Gradient $\leftarrow -\nabla_{\{\theta,\psi,\phi,\lambda,\gamma,\mathbf{Z}\}}\text{ELBO}$
15:         Update $\{\theta, \psi, \phi, \lambda, \gamma, \mathbf{Z}\}$ using Adam optimizer.
16:     end for
17:     return Optimized parameters $\{\theta, \psi, \phi, \lambda, \gamma, \mathbf{Z}\}$
18: end procedure
```

9) PREDICTION

After optimizing the hyperparameters using a training dataset, the aim is to employ LDGD to predict the labels for a new set of measurements, denoted as $\mathbf{Y}^{r*}$. This set of measurements constitutes the test set. We assume that the corresponding labels $\mathbf{Y}^{c*}$ for these new data points are absent. In essence, the test data points are employed to infer their latent features, and these latent features are subsequently decoded to determine the distribution over the corresponding labels.

The variational posterior for the latent variables $\mathbf{X}$ was introduced in Equation (20). Given a new test point $\mathbf{Y}^{r*}$, the objective is to compute the posterior distribution $p(\mathbf{X}^* \mid \mathbf{Y}^{r*}, \mathbf{Y}^r, \mathbf{Y}^c)$, which represents the belief about the latent variables after observing the new test data. This can be approximated as $q(\mathbf{X}^*) = \prod_{i=1}^{N_{test}} \mathcal{N}(\mathbf{x}_i^* \mid \boldsymbol{\mu}_*, \mathbf{s}_*\mathbb{I}_Q)$.

To find the optimum $\boldsymbol{\mu}_*$ and $\mathbf{s}_*$, the objective is to minimize the negative of the lower bound with respect to these new variational parameters. During this process, the inducing points and inducing variables are held constant. Therefore, the objective function for optimizing the variational parameters associated with a test point is formulated as follows:

$$\arg\max_{\boldsymbol{\mu}_*, \mathbf{s}_*} \left\{ \sum_d \text{ELL}^{\text{reg}}_{*,d} - \text{KL}\left(q_\phi(\mathbf{X}^*) \parallel p(\mathbf{X}^*)\right) \right\}. \quad (53)$$

Using this result, $p(\mathbf{Y}^{c*} \mid \mathbf{X}^*)$ can be effectively decoded and determined.

10) FAST LDGD

In LDGD, an iterative optimization needs to be run in the prediction or test phase, making it less suitable for real-time scenarios. In this section, we introduce Fast LDGD, in which a neural network is employed to estimate the variational parameters for the latent space directly. To this end, the variational distributions introduced in Equation (22) are modified, and the posterior $p(\mathbf{X} \mid \mathbf{Y}^r, \mathbf{Y}^c)$ is estimated with:

$$q_\phi(\mathbf{X}) = \prod_{i=1}^{N} \mathcal{N}(\mathbf{x}_i \mid \mu(\mathbf{y}_i^r), s(\mathbf{y}_i^r)\mathbb{I}_Q), \quad (54)$$

where $\mu(\mathbf{y}_i^r)$ and $s(\mathbf{y}_i^r)$ are neural network outputs with trainable parameters ϕ. The architecture of the neural network can be chosen based on the data. In this approach, the number of the model parameters depends on the continuous data dimensionality and does not depend on the number of data points. In contrast, in LDGD, the number of variational parameters in Equation (22) grows linearly with the number of training samples. Thus, this approach is beneficial when dealing with a decoding problem (classification) with a relatively larger number of data points. Our goal in the training phase is to minimize the negative of the ELBO by optimizing neural network parameters, inducing points, inducing variables' variational parameters, kernel parameters, and model parameters. By achieving this, the neural network ensures a suitable approximation to the true posterior given the continuous data. Once trained, the model can instantly provide estimates of the posterior's mean and covariance for a test point without additional optimization. This capability facilitates real-time inference.

By directly mapping from observations to the variational parameters of latent variables, the need for iterative optimization to determine μ_i and $\mathbf{s}_i$ (introduced in Equation (22)) for each test data point is bypassed. Consequently, this significantly reduces computational time, enabling real-time predictions, which is crucial in time-sensitive applications.

C. EXPERIMENTAL SETUP

The experiments were conducted using a setup of both software and hardware. The software environment included Windows as the operating system, utilizing Python 3.10 as the programming language. Key libraries included PyTorch 1.12.1 [38] and GPyTorch 1.11 [39] for implementing machine learning models.

The hardware setup featured an Intel Core i7-10700K CPU @ 3.80GHz, providing strong performance, complemented by an NVIDIA GeForce RTX 3080 GPU for handling intensive deep learning tasks. The system was equipped with 32GB of DDR4 RAM and a 1TB NVMe SSD to ensure fast data access and smooth performance.

III. EXPERIMENTS AND RESULTS

In this section, different attributes of LDGD are studied by applying it to various datasets. In particular, we examine its capability to draw low-dimensional data representations, decode accurately (classification), and generate data points. The attributes of LDGD are investigated using both synthetic and real-world datasets. By utilizing synthetic data, a comprehensive investigation of the model attributes under controlled conditions can be conducted. The real-world datasets include the Oil Flow [40], Iris [41], and MNIST datasets [42]. Through these datasets, the model's attributes are further explored to determine how they extend to handling complex datasets, thereby underscoring the unique benefits of LDGD. These benefits include its ability to infer meaningful features from high-dimensional data, address variability present in the data, and achieve optimal low-dimensional data representation.

We also compare LDGD modeling results with different variants of GPLVM. Through these comparisons, the aim is to underscore LDGD's distinctive advantages. In applying this model to classification tasks, it is evaluated across three datasets using classification metrics to demonstrate the model's robustness in class label prediction. Additionally, LDGD classification accuracy is compared with two other supervised GPLVM variants, SLLGPLVM [25] and SGPLVM [23], to illustrate its superiority in classification. The model's data generation capability is explored using the MNIST dataset, showing that LDGD can effectively replicate the complex patterns of handwritten digits. LDGD and fast LDGD are also compared with variational auto-encoders, highlighting LDGD and fast LDGD's key features over variational auto-encoders, including optimal discovery of low-dimensional representation, robustness with limited datasets, and an efficient inference process. These analyses illustrate LDGD's advantages in analyzing complex and high-dimensional datasets.

A. DATASETS

1) SYNTHETIC DATA

For the synthetic data, we utilize a "moon-like" dataset, which is a two-dimensional dataset resembling a pair of crescent moons depicted in Figure 2. This two-dimensional data is transferred into a much higher dimensional space using two different transformations. The first transformation involves a linear transformation, where the original data is multiplied by a randomly generated matrix of size $D \times 2$, with D set to 5, 15, or 20. In the second transformation, the dimension of the data is doubled by adding white noise data dimensions. For instance, if the data dimension is 5, five channels of independent white noise with unit variance are generated and appended to the data. The added dimensions do not carry class-specific information, making inference and classification tasks more challenging. For each specified dimensionality, 500 synthetic data samples were generated, divided into two classes with 250 samples per class, and labeled synthetic-10, synthetic-30, and synthetic-40.

2) OIL FLOW DATASET

The Oil Flow dataset is a high-dimensional dataset consisting of 12-dimensional feature vectors. These data points are recorded to capture various factors representing oil flow dynamics within a pipeline [40]. The dataset embodies three distinct phases of flow: horizontally stratified, nested annular, and homogeneous mixture flow. These three phases represent the conditions under which oil, gas, and water can coexist and move through a pipeline. Despite the high dimensionality of the dataset, the crucial information about the flow conditions can be effectively described by two main variables: the fraction of water and the fraction of oil in the flow. This property of the dataset makes it an extensively used benchmark for evaluating the effectiveness of dimensionality reduction techniques, showcasing their ability to simplify and interpret complex, high-dimensional data in fluid dynamics and pipeline management. A total of 1,000 samples are included in the dataset. This dataset is publicly available and can be found at [43].

3) IRIS DATASET

The Iris dataset is widely used in machine learning, covering morphological variations across three categories of iris flowers, namely Setosa, Versicolor, and Virginica [41]. This dataset is publicly available in the UCI Machine Learning Repository [44]. It comprises 150 data points distributed equally among the species, each detailing four critical features reflecting the flowers' physical characteristics, specifically petal and sepal sizes. The dataset's clear class distinctions and simple structure make it an ideal example of how our dimensionality reduction technique finds meaningful low-dimensional representation.

4) MNIST DATASET

The MNIST (Modified National Institute of Standards and Technology) dataset is popularly used in machine learning and image processing [42]. The dataset comprises 70,000 images of handwritten digits (0 through 9), and it serves as a fundamental benchmark for evaluating the performance of various machine-learning algorithms. Each image is represented in a gray-scale format with a resolution of 28×28 pixels, equating to 784 dimensions when flattened into a vector form. The dataset is publicly available and can be accessed at [42].

B. PERFORMANCE ASSESSMENT

1) SYNTHETIC DATA

a: LOWER DIMENSION REPRESENTATION

We argue that two key advantages of our model are its ability to infer meaningful information in high-dimensional data and, more importantly, quantify the validity of its probabilistic inference. To demonstrate this, LDGD is applied on the synthetic-10, synthetic-30, and synthetic-40 datasets, described in Section III-A2. We run LDGD with two different settings. In the first setting, it is assumed that Q, the latent dimension, is 2. This choice of dimension is helpful to properly visualize the inferred $\mathbf{X}$. Figure 2-B, 2-C, and 2-D respectively show inferred $\mathbf{X}$ for the synthetic-10, synthetic-30, and synthetic-40 dataset. The inferred points (posterior mean of $\mathbf{X}$) represent a rotated version of the initial data shown in Figure 2-A. Here, our pipeline converges to a similar data representation independent of the observed data dimension. The variance of the predictive distribution $p(\mathbf{Y}^c \mid X)$ serves as a measure of uncertainty in label prediction. The heatmap, employing a binary colormap, visualizes the uncertainty levels across different regions. The inferred posterior variance reflects higher confidence in areas with denser data concentrations (segments) and lower confidence in regions where these concentrations approach one another.

We use the second setting to highlight LDGD's capability to infer the latent dimension representing observed data and corresponding labels optimally. This can be done by adjusting ARD coefficients in the training step. For this setting, the synthetic-10 dataset is used, and LDGD's latent dimensions are set to 10, equal to the data dimension, with 25 inducing points for each classification and regression path. The model was trained using 80% of the dataset, selected randomly. The training process aimed to minimize Equation (26), as described in Section II-B5. Figure 2-E shows values of ELBO loss over training iterations. For the classification, the ARD coefficients indicate LDGD converges to a 2-D representation aligned with what is embedded in the data (Figure 2-F). For the regression, the ARD coefficients suggest that all dimensions are employed for data generation (Figure 2-G). The ARD coefficients in each of the two paths, along with the inferred state, demonstrate how the framework tries to reconstruct both paths through the same latent structure. The inferred $\mathbf{X}$ shows what the model requires for classification and what is needed for reconstructing data. The regression ARD coefficients display larger values for the dimensions identified by the classification ARD coefficients, suggesting that these dimensions play a significant role in data reconstruction. However, additional dimensions are also necessary. This is anticipated, as the expanded data in each dimension is a weighted sum of the original data, necessitating the involvement of other latent dimensions as well.

Subsequently, with the optimized hyperparameters, the partial ELBO introduced in Equation (53) is optimized to find the estimated posterior over latent variables for the remaining 20% of the data without any knowledge of the labels. Figure 2-I demonstrates that, even for test data with unknown labels, the model could discern the intrinsic pattern hidden within the high dimensions.

TABLE 1. Classification performance of LDGD for synthetic data.

Dataset	Precision	Recall	F1 Score
Synthetic-10	1.0	1.0	1.0
Synthetic-20	1.0	1.0	1.0
Synthetic-40	1.0	1.0	1.0

b: CLASSIFICATION RESULTS

To assess the model's performance in classification tasks, LDGD was evaluated using 5-fold cross-validation. In each fold, the parameters were optimized using 80% of the synthetic dataset's data points, with the remaining 20% serving as the test set, where labels were assumed to be unknown. Twenty-five inducing points within a 10-dimensional latent manifold were utilized for LDGD. Subsequently, the model's precision, recall, and F-measure were evaluated on the test set. These results are detailed in Table 1. The model demonstrated accurate label prediction on the test set, showcasing another critical feature of LDGD: its capability to decode labels.

2) OIL FLOW, IRIS, AND MNIST DATASET

a: LOWER DIMENSION REPRESENTATION

To investigate the LDGD application further in inferring low-dimensional representation of data, we applied it to the Oil Flow, Iris, and MNIST datasets. An ARD kernel with seven dimensions is used for the Oil Flow and Iris datasets. The model exhibits low sensitivity to the choice of latent dimension size; therefore, a latent dimension size of 7 is selected, which exceeds the dimensionality of the Iris dataset but remains below that of the Oil Flow dataset. Ten inducing points are assumed for both the classification and regression paths. The parameters are optimized using 80% of the data along with their labels, and the latent space for the remaining 20% is inferred without label information. The results indicate that the trained ARD coefficients suggest an optimal use of about one dimension for the Iris dataset and two dimensions for the Oil Flow dataset (see Figure 3). It can be observed that there are different levels of certainty in each axis; for instance, in Figure 3-B, much higher confidence is seen in the x-axis, corresponding to the 5th element in ARD space, indicating its significant role in the model's predictions. On the other hand, the second-largest ARD coefficient is less significant, leading to lower confidence in the y-axis. For Iris, the dimension of $\mathbf{X}$ is larger than the observed data. Though this is not the case the framework is designed for, it is observed that ARD only keeps two coefficients and pushes other coefficients toward zero. Though this does not reflect any attributes of the LDGD, it shows the robustness of the pipeline and its principled representation of the latent data.

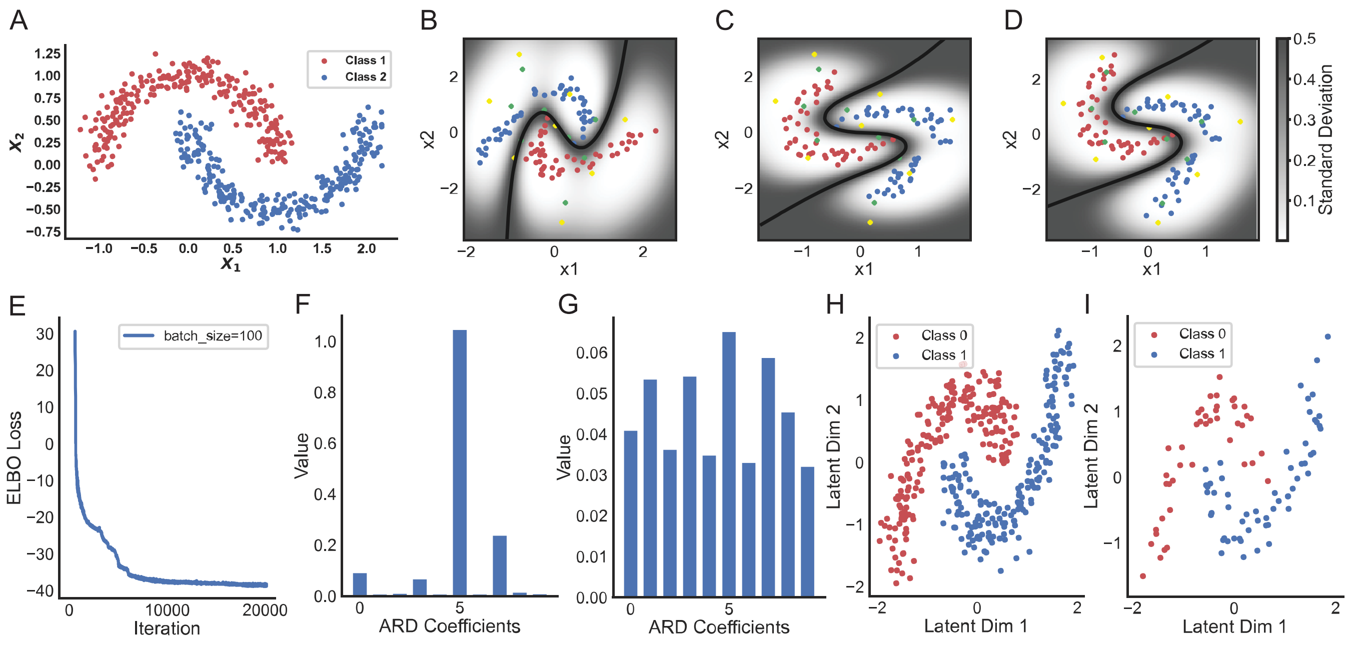

FIGURE 2. **Visualization of Dimensionality Reduction on Synthetic Data. (A) Displays the initial two dimensions of the moon-like dataset. (B-D) Illustrates the latent space heatmap representation across different synthetic data dimensions: 10 dimensions (B), 20 dimensions (C), and 40 dimensions (D). Red and blue points represent class 1 samples ($y^c = 0$) and class 2 samples ($y^c = 1$), respectively. Green crosses indicate classification-inducing points, while yellow crosses denote regression-inducing points, with five inducing points used for each. The green data points are more uniformly distributed over the space as it is constructing the $\mathbf{Y}^r$. The yellow ones are aligned on two sides of the decision boundary as they are more needed for a correct classification. The heatmap visualizes the model's uncertainty level (posterior variance). (E) Shows the training curve (ELBO loss) for synthetic data with ten dimensions, where the latent space is also set to 10 dimensions. (F-G) Depicts the ARD coefficients for the classification kernel (F) and the regression kernel (G). The trained coefficients highlight that the model selects two dimensions to represent a 10-dimensional space in a lower-dimensional setting for decoding labels and employs almost all dimensions to reconstruct data in the original space. (H) displays a scatter plot of the training points in the lower-dimensional space for the two most dominant dimensions, while (I) shows the scatter plot for test points where the labels are unknown.**

Given the MNIST dataset's larger feature space, we used 20 dimensions for the latent space and 150 inducing points for each path. The model utilized almost all the available dimensions for this dataset, highlighting the necessity of high dimensions to classify and reconstruct such a large dataset. LDGD's representation in 2-D may not exhibit as much distinct separation as T-SNE due to its generative nature and probabilistic embeddings that prioritize capturing the global data structure over local neighborhood relationships.

In parallel, we applied PCA, t-SNE, GPLVM, Bayesian GPLVM, FGPLVM, SGPLVM, and SLLGPLVM to the Oil Flow and Iris datasets. The latent variable's dimensions were fixed to two across all models for fair comparisons. SGPLVM, SLLGPLVM, and LDGD utilize labels during the training process, while others are unsupervised and agnostic to the labels. Figure 4 illustrates the latent representations generated by LDGD compared to those obtained using other models. The Iris dataset, being less complex, allowed almost all models to find a lower-dimensional representation that demonstrates a clear distinction between species. So, LDGD's performance is similar to other models. For the Oil Flow dataset, LDGD and SLLGPLVM effectively separate the three flow phases, while some other methods exhibit partial overlap between different categories. Unlike SLLGPLVM, LDGD incorporates inducing points within its pipeline, which allows LDGD to scale better than SLLGPLVM. Consequently, a direct comparison for the MNIST dataset is not feasible since SLLPLVM is ill-equipped to handle larger datasets.

b: CLASSIFICATION RESULTS

Here, our aim is to demonstrate the model's ability to decode or classify real-world datasets. We have applied it to the Iris and Oil Flow with seven latent dimensions and ten inducing points. LDGD was then applied with 20 latent variables and 150 inducing points for classification and regression on the MNIST dataset. The evaluation criteria included classification accuracy, precision, recall, and F1 score. The results are provided in Table 2. As can be seen, the model performance is as good as the state of the arts in the iris and oil dataset, and the performance in the MNIST dataset is not far from the other image classifier methods like convolutional neural network (CNN) methods but comparable to them [45]. The classification results in datasets with different input dimensions and different data points further demonstrate the model potential in the feature extraction and classification step.

To quantitatively compare our model's classification capabilities, we benchmarked its performance against several state-of-the-art models. While our model and others perform well on the simpler Iris dataset, LDGD stands out for its

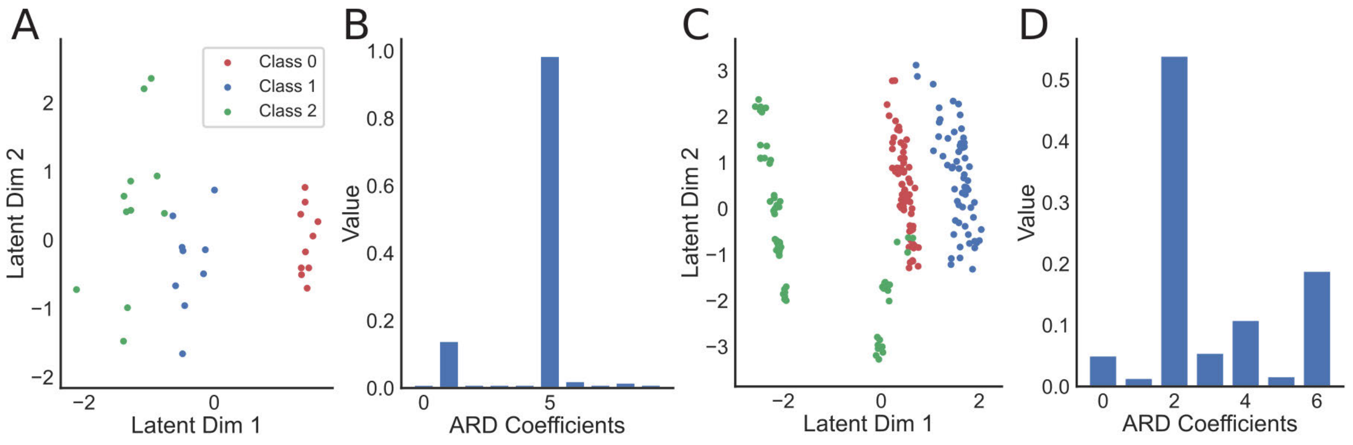

FIGURE 3. Comparative analysis of latent space representation in Iris and Oil Flow dataset. (A) reveals the most dominant latent dimension for the Iris dataset, identified through ARD coefficients. (B) illustrates the scatter plot of the two dominant dimensions in the latent space for the Iris dataset, highlighting the data's intrinsic clustering. The vertical and horizontal error bars show the variance at each data point in latent space. (C) displays the most dominant latent dimension for the Oil dataset, as determined by ARD coefficients. (D) presents the scatter plot of the two dominant dimensions in the latent space for the Oil Flow dataset, showcasing its unique distribution.

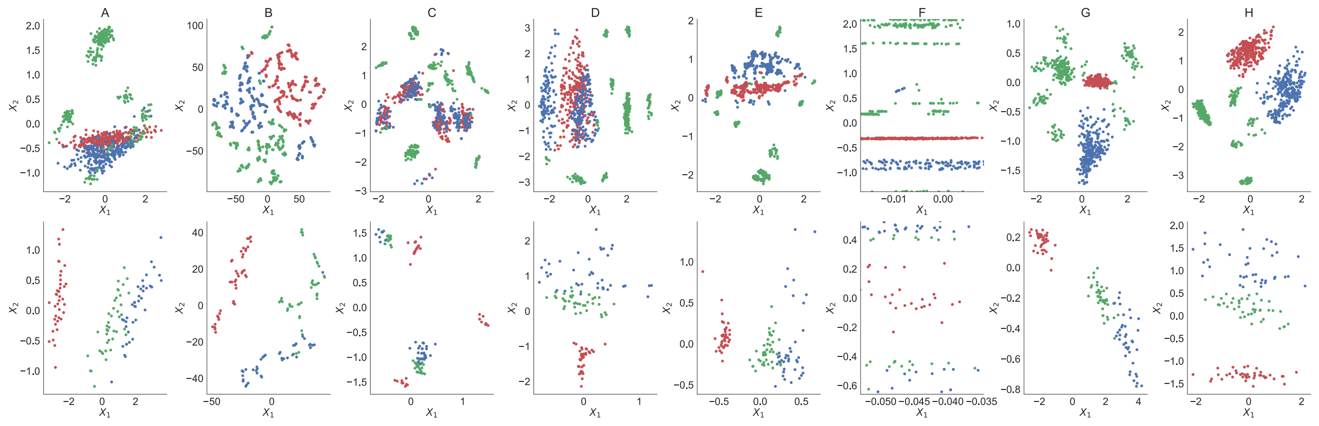

FIGURE 4. Two-dimensional visualization of dataset projections using various dimensionality reduction techniques. The top row displays projections of the Oil Flow dataset, while the bottom row shows the Iris dataset. Techniques used include (A) PCA, (B) t-SNE, (C) GPLVM, (D) Bayesian GPLVM, (E) FGPLVM, (F) SGPLVM, (G) SLLGPLVM, and (H) LDGD. It was observed that both SLLGPLVM and LDGD inference in low dimensions reflect the class labels, resulting in separate regions for different data classes. With LDGD, a one-dimensional representation of data can be achieved. Thus, a better reduction rate might be obtained using LDGD.

TABLE 2. Classification performance of LDGD for Iris, Oil Flow, and MNIST datasets.

Dataset	Accuracy	Precision	Recall	F1 Score
Iris	1.00	1.00	1.00	1.00
Oil Flow	0.99	1.00	0.99	0.99
MNIST	0.95	0.94	0.93	0.94

TABLE 3. Classification performance of LDGD for the Oil Flow dataset compared to other models.

Model	Accuracy	Precision	Recall	F1 Score
LDGD	0.99	1.00	0.99	0.99
SGPLVM	0.93	0.92	0.91	0.92
SLLGPLVM	0.95	0.94	0.93	0.94

unique ability to handle the complexity of the MNIST dataset. Consequently, our comparison focuses on the Oil Flow dataset to provide a fair comparison of LDGD's performance with its competitors.

Table 3 summarizes the classification results, where LDGD consistently outperforms other models across all measures. This confirms LDGD's effectiveness in extracting latent features and accurately predicting labels. The model's superior classification performance results from its adaptive feature extraction capabilities. These empirical findings highlight LDGD's dual role as both an efficient tool for dimensionality reduction and a powerful discriminative model. The model's achievement of high classification metrics across various datasets underscores its adaptability and potential for use in numerous practical machine-learning scenarios, even with larger datasets. Our results for MNIST, a comparatively large dataset for Gaussian process-based models, demonstrate LDGD's scalability. The LDGD model surpasses both SLLGPLVM and SGPLVM in terms of scalability. The scalability is attributed mainly to integrating

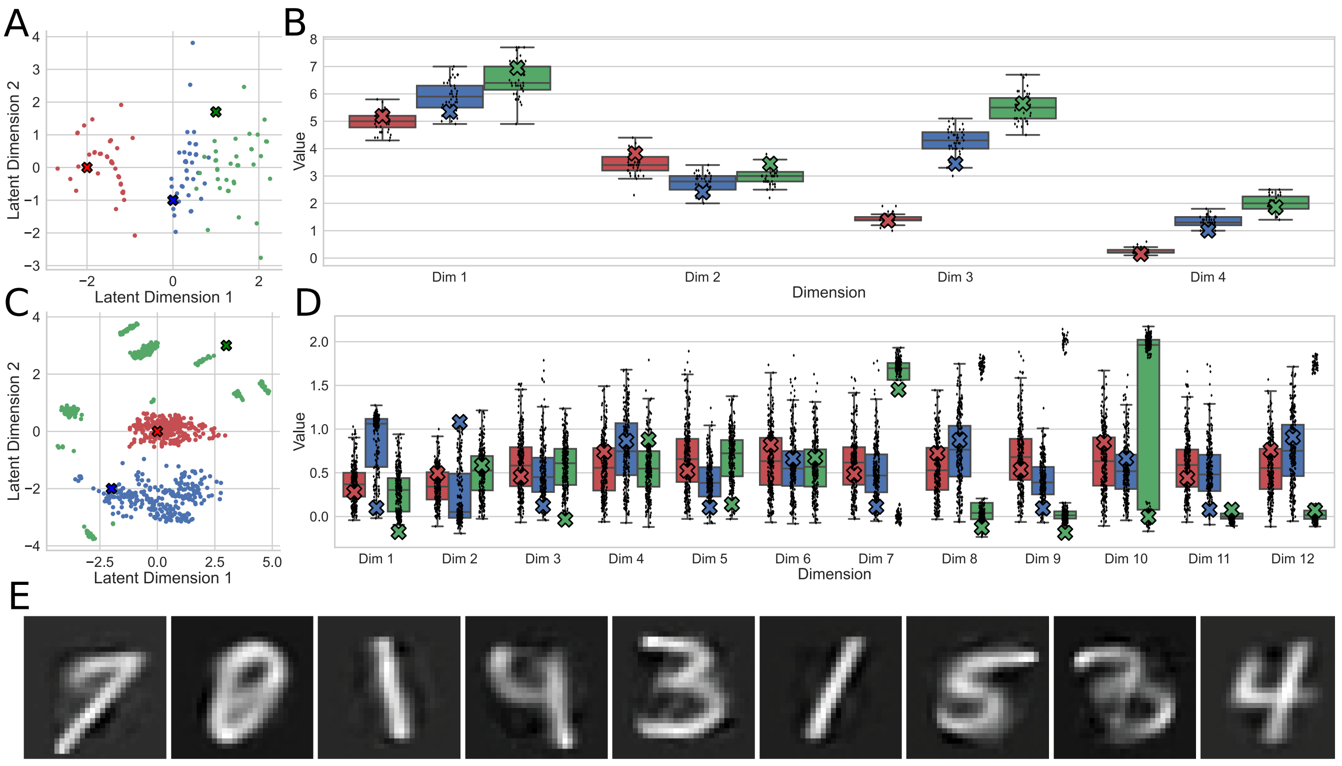

FIGURE 5. LDGD Data Generation Analysis. (A-D) These figures present scatter plots of data points in the 2D latent space for the Iris (A) and the Oil Flow dataset (C). For each dataset, a sample point near each cluster is randomly selected (marked with a cross) to illustrate the model's generative capabilities. The corresponding high-dimensional reconstructions of these starred points are shown for the Iris dataset (B) and the Oil Flow dataset (D), alongside the real values of other points. (E) displays nine randomly chosen test points in the latent space for the MNIST dataset and their reconstructed images through the model's generative path. The generated images properly reconstruct corresponding digits.

batch training, which enables efficient handling of large datasets without compromising the model's performance or accuracy.

c: PROCESSING TIME

All the training for the models was done using the experimental setup described in Section II-C, with the SLLGPLVM and SGPLVM executed on MATLAB 2022 [46]. In terms of processing time, the collective analyses described below suggest its computational efficiency is on par with other state-of-art models and in some cases even more efficient. The execution time for training the LDGD model on 200 samples from the oil-flow dataset is about 327 seconds for 3000 iterations, compared to 346 seconds for SGPLVM and 3100 seconds for SLLGPLVM. As the data size increases to 800 samples, the training time for SLLGPLVM becomes significantly longer going beyond 216000 seconds, whereas LDGD processing time will grow to 542 seconds suggesting a reasonable training time. This timing suggests the computational efficiency of Gaussian process models, though it is still slower than some parametric models [47]. The balance between accuracy and processing time observed in our proposed model makes it a viable option for applications requiring high precision without excessively long training times.

d: LDGD DATA GENERATION

So far, our focus has been on inferring the latent space and then utilizing LDGD for label prediction. It is noteworthy that LDGD is capable of generating continuous samples given a label. Figure 5 displays three sample data points, each selected randomly near a class in the latent space within the Iris and Oil Flow datasets. As observed, the reconstructed samples in high-dimensional space lie close to the actual data points. This demonstrates how LDGD training effectively balances the label and continuous paths, ensuring that the generated data accurately represents its respective classes. Additionally, Figure 5-C illustrates nine sample digits generated randomly from the MNIST dataset, highlighting the model's ability to produce samples from more complex data.

C. LDGD, FAST LDGD, AND VARIATIONAL AUTO-ENCODERS

While LDGD might resemble a variational auto-encoder (VAE), fundamental differences between our proposed framework and VAE models must be highlighted further. In VAEs, the latent manifold has Gaussian priors, similar to LDGD and fast LDGD, where the posterior over latent variables are approximated using a Gaussian distribution. However, the distinctions lie in the relationship between the

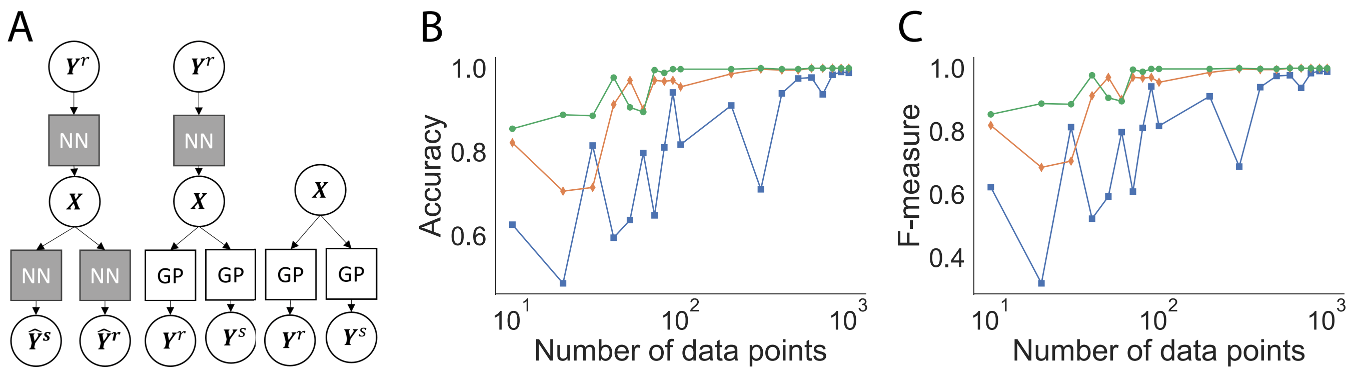

FIGURE 6. **Comparison of LDGD, Fast LDGD, and VAE Models. (A) shows the graphical models for VAE, fast LDGD, and LDGD from left to right. (B, C) illustrate the relationship between the number of training samples (x-axis) and the performance metrics on the test set, specifically accuracy (B) and F-measure (C), where the blue line represents VAE performance, red indicates LDGD and green denotes fast LDGD performance.**

latent manifold, labels, and high-dimensional data. LDGD employs a probabilistic framework, whereas VAEs utilize a deterministic framework (neural networks). Fast LDGD occupies a middle ground, inferring $\mathbf{X}$ through a neural network while leveraging a Gaussian Process (GP) model for label prediction. Another key difference resides in the loss functions used: VAEs incorporate mean squared error (MSE), categorical entropy, and Kullback-Leibler (KL) divergence, whereas LDGD relies on the ELBO. Figure 6.A shows the probabilistic graphs for these three methods. As observed in LDGD, the objective is to find the posterior over $\mathbf{X}$ given $\mathbf{Y}^r$ and $\mathbf{Y}^c$.

LDGD has multiple advantages over the VAE model, making it a more suitable model for analyzing high-dimensional data, specifically when the size of the dataset is limited. Using LDGD, the optimal dimension of the latent process can be identified as a part of learning. In contrast, the number of dimensions must be tuned as a hyperparameter when using the VAE model. Inferencing and training in LDGD are not data-greedy processes; the model can be built with much smaller data compared to what is needed for VAE model training. Compared to a point estimate in VAE, Bayesian inferencing helps us have higher-order statistics, such as confidence in our prediction of a label or data point, giving us a better sense of our predictions.

Another key advantage is the inherent robustness of LDGD against overfitting, a common pitfall in many machine learning models. LDGD's use of Bayesian inferencing inherently guards against overfitting, making LDGD particularly suited for dealing with small datasets. This framework doesn't merely fit the model to the data; it enables the model to correctly infer and adapt to the data, even when the dataset is small. To assess the model's performance in such scenarios, we have used the synthetic-20 dataset. Different amounts of training data with the same amount of test data were used to compare VAE, LDGD, and fast LDGD. As shown in Figure 6-B and 6-C, for small datasets, LDGD and fast LDGD's performance in classification is significantly better. For large datasets, the performance is comparable. These advantages collectively position LDGD as a mere alternative and a significant advancement over traditional encoder-decoder models, especially in high-dimensional data analysis.

IV. DISCUSSION

In this research, different modeling steps of LDGD were developed, including its training, inferencing, and decoding (classification). New approaches were introduced in developing the framework, including a shared latent variable with doubly stochastic variational inference with trainable inducing points. To assess the model's different attributes, the framework was applied to the Synthetic, Iris, and Oil Flow datasets.

The modeling results in both synthetic and real-world data suggest LDGD's capability to correctly infer latent space and identify the latent space's proper dimension. The classification results using LDGD are on par with the state-of-the-art models, and in some of the data points, it even surpasses other approaches.

Different attributes and potential advantages of LDGD were highlighted. We argued its capabilities in identifying the optimal dimension of the latent process. For instance, in synthetic data, it was demonstrated that the framework would suggest a 2-D dimensional latent space, which matches the intrinsic dimensions of the original data. This can be done using an ARD kernel where its coefficients are trained during the model training. This is a remarkable feature of the model, which is not present in other algorithms such as VAEs. LDGD's expressiveness and ability to classify data labels precisely were also highlighted. It was demonstrated that LDGD can be a highly effective classifier for the Iris and Oil Flow datasets, surpassing state-of-the-art approaches. LDGD can be viewed as a pipeline with adaptive feature extraction and a classifier with features capable of reconstructing the data. Our results demonstrated its capability to process larger datasets, such as MNIST, efficiently, which was achieved by adopting batch training, the inducing point concept, and a doubly variational strategy. Specifically, this approach highlighted LDGD's scalability to larger datasets, a feature distinctly lacking in models like SGPLVM and SLLGPLVM. The generative prowess of LDGD was also explored, highlighting its ability to create data from high-dimensional spaces. Our experiments confirmed that LDGD can adeptly generate samples from the Iris, Oil Flow, and MNIST datasets. The generated samples from MNIST were remarkably coherent.

A couple of challenges with LDGD were also observed. A core challenge is hyperparameter selection, such as determining the number of inducing points. As with many in machine learning, the model's performance hinges on hyperparameter choices. In Gaussian Process-based models, the number of inducing points is critical; too few may lead to underfitting and inadequate data complexity representation, while too many can escalate computational demands and risk overfitting. Finding the optimal number often requires a dataset-specific balance, posing practical challenges. Although LDGD improves scalability, it may struggle with extremely large or high-dimensional datasets due to the inherent complexity of Bayesian models and Gaussian Processes, limiting its feasibility in some real-time applications.

LDGD's reliance on label quality for dimensionality reduction means its performance is contingent on label accuracy. The model's efficiency may diminish in cases of scarce, inaccurate, or biased labels. Despite its proficiency in uncertainty quantification, the interpretability of complex models like LDGD can be daunting, especially for users without a machine learning background. The contribution of various modeling steps to create the outcome might make the pipeline hard to digest. Furthermore, while LDGD shows promise for particular datasets, its performance across diverse data types, such as time series, images, or text, has not been thoroughly tested, leaving its applicability in these areas yet to be fully established.

Several improvements are suggested to overcome these limitations and extend the applicability of LDGD. These include the development of methods for the automatic selection of inducing points, such as Bayesian optimization and cross-validation approaches, as well as extensions of the model to handle time series data. Several modifications are suggested to enhance the LDGD model for time series analysis. Firstly, incorporate temporal dynamics by integrating autoregressive components, state space models, or using time series-specific kernels in Gaussian Processes [48], [49], [50], [51]. Secondly, implementing methods to handle sequential data, possibly using recurrent neural network structures like LSTMs [52] or GRUs [53] within the LDGD framework. Thirdly, explore time-variant inducing points that adapt to changes in time series data, improving the model's ability to capture temporal dynamics. Parallel computing, GPU acceleration, and sparse variational frameworks are also recommended to enhance algorithmic efficiency and scalability.

In general, it is essential to establish standard benchmarks and evaluation methodologies to advance the field of dimensionality reduction and ensure the efficacy of techniques like the LDGD model. Additionally, it's crucial to continuously benchmark the enhanced model against the latest state-of-the-art models in dimensionality reduction. This ongoing comparison will ensure the model's competitiveness and help identify potential areas for further improvement.

Future work will focus on extending LDGD for specific applications, such as datasets recorded during neuroscience experiments, which are generally small in size and exhibit complex variability [54]. We also aim to enhance algorithmic efficiency and conduct robustness and generalizability studies. In summary, the LDGD model offers promising advancements in dimensionality reduction and classification, with wide-ranging applications and potential for significant contributions to various fields. However, realizing its full potential requires continuous research and development to address its current limitations and expand its applicability in addressing complex real-world problems.

V. CONCLUSION

In this research, we showcased the utilities of non-parametric models for dimensionality reduction, feature discovery, and decoding. Despite the emergence of powerful machine-learning tools, there is still a lack of principled tools capable of dealing with intrinsic stochasticity and variability in data or scenarios where the size of available data is limited. The LDGD framework addresses these needs and achieves comparable or even higher prediction accuracy and generative capability. With LDGD, data from domains such as neuroscience, which are generally complex and limited in size, can be studied effectively. The ideas in LDGD can be extended to more realistic data, such as time series, thus requiring further model development, which is the core of our future research.

APPENDIX A GAUSSIAN PROCESS REGRESSION

Gaussian processes (GP) offer a powerful non-parametric approach for regression and classification tasks [29]. A Gaussian process is a collection of random variables, with the property that any finite subset of these variables has a joint Gaussian distribution. GP defines a distribution over a function defined by:

$$p(\mathbf{f} \mid \mathbf{X}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}_{NN}), \tag{55}$$

where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ are input vectors with $\mathbf{x}_i \in \mathbb{R}^d$, $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]$ represents latent variables, called function values in GP context, at the input points with mean $\boldsymbol{\mu} = [m(\mathbf{x}_1), \ldots, m(\mathbf{x}_N)]$ and a $N \times N$ covariance matrix $\mathbf{K}_{NN}$. Each element in the ith row and jth column is given by $K_{ij} = k_\theta(\mathbf{x}_i, \mathbf{x}_j)$ for $i, j \in \{1, \ldots, N\}$, where k is a positive definite kernel constructing elements of the covariance matrix $\mathbf{K}_{NN}$ and θ defines the kernel's free parameters. In practice, with no prior observation, the mean of the function is assumed to be zero, i.e., $m(\mathbf{x}) = \mathbf{0}$.

Consider a dataset $d = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, where $y_i \in \mathbb{R}$ represents the noisy observations at each input location $\mathbf{x}_i$ within the domain of interest, encapsulating the underlying process; we aim to model using a Gaussian process. The generation of y_i is modeled as follows:

$$y_i = f(\mathbf{x}_i) + \epsilon, \tag{56}$$

where ϵ is considered as Gaussian noise, defined by $\epsilon \sim \mathcal{N}(0, \sigma_y^2)$. The likelihood of observing y_i given $\mathbf{x}_i$ is defined by

$$p(y_i \mid \mathbf{x}_i) = \mathcal{N}(y_i \mid f(\mathbf{x}_i), \sigma_y^2). \tag{57}$$

Furthermore, given the GP model, the joint distribution of $\mathbf{y} = [y_1, \ldots, y_N]$ given $\mathbf{X}$ is defined by

$$p(\mathbf{y} \mid \mathbf{X}) = \mathcal{N}(\mathbf{y} \mid \mathbf{0}, \mathbf{K}_{NN} + \sigma_y^2 \mathbb{I}_N). \tag{58}$$

To train the model, the kernel parameters (θ) need to be estimated, which can be found by solving

$$\arg\max_{\theta} \log p(\mathbf{y} \mid \mathbf{X}). \tag{59}$$

For a set of test points $\mathbf{X}^* \equiv [\mathbf{x}_1^*, \ldots, \mathbf{x}_{N*}^*]$, the objective is to find the predictive distribution $p(\mathbf{y}^* \mid \mathbf{y}, \mathbf{X}, \mathbf{X}^*) = \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)$, where the mean and covariance are defined by

$$\boldsymbol{\mu}^* = \mathbf{K}_{*N}\mathbf{K}_{NN}^{-1}\mathbf{y}, \tag{60}$$
$$\boldsymbol{\Sigma}^* = \mathbf{K}_{**} - \mathbf{K}_{*N}(\mathbf{K}_{NN} + \sigma_y^2 \mathbb{I}_N)^{-1}\mathbf{K}_{N*}. \tag{61}$$

Here, $\mathbf{K}_{**}$ is the covariance matrix between the test points, and $\mathbf{K}_{*N}$ is the cross-covariance matrix between the test points and the training points.

The formulation requires the inversion of an $N \times N$ matrix to derive the predictive distribution, resulting in a computational complexity of $\mathcal{O}(N^3)$. This computation becomes expensive in datasets even with a moderate number of data points. A solution that addresses this challenge is based on variational inducing points proposed by [18]. In this approach, M data points $\mathbf{Z} \equiv \{\mathbf{z}_i\}_{i=1}^{M}$, where $\mathbf{z}_i \in \mathbb{R}^d$, is introduced in the input space. These M data points, where $M \ll N$, replace the N original data points used in deriving the covariance matrix. Inducing points $\mathbf{Z}$ can be chosen as fixed points or model-free parameters [18]. This research considers $\mathbf{Z}$ as free parameters with random initial values. It is further assumed that for each inducing point, there is a corresponding inducing variable u_i. Induced variables $\mathbf{u} \equiv [u_1, \ldots, u_M]$ follow the distribution $p(\mathbf{u} \mid \mathbf{Z}) = \mathcal{N}(\mathbf{u} \mid \mathbf{0}, \mathbf{K}_{MM})$, where $\mathbf{K}_{MM}$ is an $M \times M$ matrix. This matrix is generated using the same kernel function $k_\theta(z_i, z_j)$ as in the GP model, applied to the inducing points. The conditional distribution of $\mathbf{f}$ given $\mathbf{u}$ is:

$$p(\mathbf{f} \mid \mathbf{u}; \mathbf{Z}, \mathbf{X}) = \mathcal{N}(\mathbf{f} \mid \boldsymbol{\mu}_f, \boldsymbol{\Sigma}_f), \tag{62}$$

where $\boldsymbol{\mu}_f = \mathbf{K}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{u}$ and $\boldsymbol{\Sigma}_f = \mathbf{K}_{NN} - \mathbf{K}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{K}_{MN}$. Using the inducing points, predictions for new points can be drawn with a much smaller dataset, significantly reducing the computational cost for the covariance inverse. However, to use $\mathbf{Z}$ for the prediction step, these points need to be estimated first. With the inducing points, the free parameters of the GP model increase to include both θ and $\mathbf{Z}$. The maximum likelihood estimate of θ and $\mathbf{Z}$ is found through the marginal distribution of $\mathbf{y}$, defined by:

$$\begin{aligned} p(\mathbf{y} \mid \mathbf{X}; \mathbf{Z}, \theta) &= \int p(\mathbf{y}, \mathbf{f}, \mathbf{u} \mid \mathbf{X}; \mathbf{Z}, \theta)\, d\mathbf{f}\, d\mathbf{u} \\ &= \int p(\mathbf{y} \mid \mathbf{f}; \theta) p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}; \mathbf{Z}, \theta) p(\mathbf{u}; \mathbf{Z}, \theta) \\ &\quad \times d\mathbf{f}\, d\mathbf{u}, \end{aligned} \tag{63}$$

where $\mathbf{f}$ and $\mathbf{u}$ are sets of latent variables of length N and M, respectively. To determine θ and $\mathbf{Z}$ without needing to invert the large matrix $\mathbf{K}_{NN}$, a variational approach is employed by introducing the variational distribution $q(\mathbf{f}, \mathbf{u})$. The distribution $q(\mathbf{f}, \mathbf{u})$ is defined as $q(\mathbf{u})q(\mathbf{f} \mid \mathbf{u})$, where $q(\mathbf{u}) = \prod_{j=1}^{M} \mathcal{N}(u_j; m_j, s_j)$ and $q(\mathbf{f} \mid \mathbf{u}) = p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}; \mathbf{Z}, \theta)$, as already defined in Equation (62). With the q distribution, a lower bound on the marginal distribution of $\mathbf{y}$ can be derived using Jensen's inequality, which becomes:

$$\begin{aligned} \log p(\mathbf{y} \mid \mathbf{X}; \mathbf{Z}, \theta) \geq\ & \mathbb{E}_{q(\mathbf{f},\mathbf{u})}[\log p(\mathbf{y} \mid \mathbf{f}; \theta)] \\ & - \mathrm{KL}[q(\mathbf{u}) \parallel p(\mathbf{u}; \mathbf{Z}, \theta)], \end{aligned} \tag{64}$$

where $\mathbf{Z}$ and θ, along with m_j, s_j for $j = 1, \ldots, M$, are found to maximize the lower bound. The first term on the right-hand side of the Equation 64 encourages the model to fit the data well, while the second term regularizes the model by penalizing deviations of the variational distribution from the prior. It is worth noting that the inversion of the $\mathbf{K}_{NN}$ matrix is no longer needed in prediction and parameter estimation steps.

For N_t test points $\mathbf{X}^* \equiv \left[x_1^*, \ldots, x_{N_t}^*\right]$, the predictive distribution is defined by $p(\mathbf{f}^* \mid \mathbf{X}, \mathbf{y}, \mathbf{X}^*)$. Given the inducing points, this distribution can be written as:

$$p(\mathbf{f}^* \mid \mathbf{X}, \mathbf{y}, \mathbf{X}^*) = \int p(\mathbf{f}^*, \mathbf{u} \mid \mathbf{X}, \mathbf{y}, \mathbf{X}^*)\, d\mathbf{u}. \tag{65}$$

Note that in the equation, for clarity, its parameters–$\mathbf{Z}$ and θ–are omitted. The exact posterior $p(f^*, \mathbf{u} \mid \mathbf{X}, \mathbf{y}, \mathbf{X}^*)$ is estimated with the updated variational distribution $q(f^*, \mathbf{u})$. Thus, the predictive distribution can be approximated as:

$$\begin{aligned} p(\mathbf{f}^* \mid \mathbf{X}, \mathbf{y}, \mathbf{X}^*) &\approx \int q(\mathbf{f}^*, \mathbf{u})\, d\mathbf{u} \\ &= \int p(\mathbf{f}^* \mid \mathbf{u}) q(\mathbf{u})\, d\mathbf{u}. \end{aligned} \tag{66}$$

These distributions have already been derived during the model training. They are defined by:

$$q(\mathbf{u}) \sim \mathcal{N}(\mathbf{u} \mid \mathbf{m}, \mathbf{S}), \tag{67}$$
$$p(\mathbf{f}^* \mid \mathbf{u}) \sim \mathcal{N}\left(\mathbf{f}^* \mid \boldsymbol{\mu}_{f*}, \boldsymbol{\Sigma}_{f*}\right). \tag{68}$$

where $\mathbf{m} = [m_1, \ldots, m_M]$ and $\mathbf{S}$ is an $M \times M$ diagonal matrix. The mean and covariance matrix of this predictive distribution are defined by $\boldsymbol{\mu}_{\mathbf{f}^*} = \mathbf{K}_{*M}\mathbf{K}_{MM}^{-1}\mathbf{u}$ and $\boldsymbol{\Sigma}_{\mathbf{f}^*} = \mathbf{K}_{**} - \mathbf{K}_{*M}\mathbf{K}_{MM}^{-1}\mathbf{K}_{M*}$, respectively. The diagonal elements of $\mathbf{S}$ are $\{s_i\}_{i=1}^{M}$, which are the variational free parameters that are optimized during training. The approximated integral in Equation (66) has a closed form, which is a Gaussian with the following mean and covariance:

$$\boldsymbol{\mu}^* = \mathbf{K}_{*M}\mathbf{K}_{MM}^{-1}\mathbf{m}, \tag{69}$$

$$\boldsymbol{\Sigma}^* = \mathbf{K}_{**} - \mathbf{K}_{*M}\mathbf{K}_{MM}^{-1}(\mathbf{K}_{MM} - S)\mathbf{K}_{MM}^{-1}\mathbf{K}_{M*}. \tag{70}$$

APPENDIX B
EXPLORING THE CONNECTION BETWEEN PPCA AND GPLVM

The GPLVM is developed based on the probabilistic principal component analysis (PPCA) framework. In PPCA, the observed data vector $\mathbf{y}_i$ for each observation i is modeled as a linear transformation of the corresponding latent variable $\mathbf{x}_i$ with additive Gaussian noise $\boldsymbol{\epsilon}_i$:

$$\mathbf{y}_i = \mathbf{W}\mathbf{x}_i + \boldsymbol{\epsilon}_i, \tag{71}$$

where $\mathbf{y}_i \in \mathbb{R}^D$ denotes the observations, $\mathbf{x}_i \in \mathbb{R}^Q$ represents the low-dimensional representation, and $\mathbf{W} \in \mathbb{R}^{D\times Q}$ is the transformation matrix mapping from the low-dimensional space to the high-dimensional space. The prior distribution over $\mathbf{X}$ is assumed to be Gaussian with a zero mean and an identity covariance matrix:

$$p(\mathbf{X}) = \prod_{i=1}^{N} \mathcal{N}(\mathbf{x}_i \mid \mathbf{0}, \mathbb{I}_Q). \tag{72}$$

Consequently, the likelihood of $\mathbf{Y}$ given the transformation matrix $\mathbf{W}$ is:

$$p(\mathbf{Y} \mid \mathbf{W}) = \prod_{i=1}^{N} \mathcal{N}(\mathbf{y}_i \mid \mathbf{0}, \mathbf{C}), \tag{73}$$

with the covariance matrix $\mathbf{C}$ defined as:

$$\mathbf{C} = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbb{I}_D. \tag{74}$$

In the dual formulation of PPCA, the model is considered from the perspective of the observed data:

$$\mathbf{y}_{:,d} = \mathbf{X}\mathbf{w}_{:,d} + \boldsymbol{\epsilon}_d, \tag{75}$$

where $\mathbf{X} \in \mathbb{R}^{N\times Q}$ represents the low-dimensional representation of all samples, $\mathbf{y}_{:,d}$ is the dth feature in high-dimensional space for all samples, and $\mathbf{w}_{:,d}$ is the dth column of $\mathbf{W}$, which is the linear transformation matrix. The prior distribution over $\mathbf{W}$ is assumed to be Gaussian with a zero mean and an identity covariance matrix:

$$p(\mathbf{W}) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{w}_{:,d} \mid \mathbf{0}, \mathbb{I}_Q). \tag{76}$$

Here, the likelihood of $\mathbf{Y}$ given the latent variables $\mathbf{X}$ is:

$$p(\mathbf{Y} \mid \mathbf{X}) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{y}_{:,d} \mid \mathbf{0}, \mathbf{K}), \tag{77}$$

where $\mathbf{K}$ is the covariance matrix reflecting a linear relationship to $\mathbf{X}$:

$$\mathbf{K} = \mathbf{X}\mathbf{X}^T + \sigma^2\mathbb{I}_N. \tag{78}$$

To accommodate the nonlinear relationships present in many real-world datasets, the dual PPCA formulation can be extended using kernel functions $k_\theta(\mathbf{x}, \mathbf{x}')$. This formulation enables the covariance matrices to be expressed through kernel functions as follows:

$$\mathbf{K} = \mathbf{K}_{NN} + \sigma^2\mathbb{I}_N, \tag{79}$$

where $\mathbf{K}_{NN}$ denotes the covariance matrix, with each element k_{ij} in the ith row and jth column calculated by the kernel function $k_\theta(\mathbf{x}_i, \mathbf{x}_j)$, reflecting the covariance between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ in the input space. The incorporation of kernel functions into the PPCA model gives rise to the Gaussian Process Latent Variable Model (GPLVM), allowing for the capture of nonlinear relationships within the data. This enhancement transforms the model into a more adaptable framework, capable of accommodating the data's inherent complexity, as described by:

$$p(\mathbf{Y} \mid \mathbf{X}) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{y}_{:,d} \mid \mathbf{0}, \mathbf{K}_{NN} + \sigma^2\mathbb{I}_N), \tag{80}$$

where $\mathbf{Y}$ represents the high-dimensional data, $\mathbf{X}$ denotes the latent low-dimensional representation, and $\mathbf{K}_{NN}$ is the covariance matrix derived from the kernel function, capturing the nonlinearities in the data. Thus, GPLVM extends the linear dimensionality reduction approach of PPCA to a nonlinear setting, offering a more powerful tool for uncovering the underlying structure in complex datasets.

APPENDIX C
EVIDENCE LOWER BOUND FOR GPLVM

In this appendix, it is demonstrated that maximizing the ELBO (Equation 26) is equivalent to minimizing the KL divergence between the true posterior distribution and the variational posterior distribution. Equations (19) and (23) depict LDGD's true and approximated posteriors, respectively. Let's denote the true posterior as $\hat{P} \triangleq p(\mathbf{F}^r, \mathbf{F}^c, \mathbf{U}^r, \mathbf{U}^c, \mathbf{X} \mid \mathbf{Y}^r, \mathbf{Y}^c)$ and approximated posterior as $\hat{Q} \triangleq q(\mathbf{F}^r, \mathbf{F}^c, \mathbf{U}^r, \mathbf{U}^c, \mathbf{X}) = q(\mathbf{F}^c, \mathbf{U}^c)q(\mathbf{F}^r, \mathbf{U}^r)q_\phi$.The process begins by computing the KL divergence between the exact posterior and the approximation:

$$KL(\hat{Q} \parallel \hat{P}) = \int \hat{Q} \log \frac{\hat{Q}}{\hat{P}} d\mathbf{F}^r\, d\mathbf{F}^c\, d\mathbf{U}^r\, d\mathbf{U}^c\, d\mathbf{X} \tag{81}$$

The posterior distribution of LDGD, given the observed data $\mathbf{Y}^r$ and $\mathbf{Y}^c$, the latent functions $\mathbf{F}^r$ and $\mathbf{F}^c$, the inducing variables $\mathbf{U}^r$ and $\mathbf{U}^c$, and the latent features $\mathbf{X}$, is defined as:

$$\hat{P} = \frac{p(\mathbf{F}^r, \mathbf{F}^c, \mathbf{U}^r, \mathbf{U}^c, \mathbf{X}, \mathbf{Y}^r, \mathbf{Y}^c)}{p(\mathbf{Y}^r, \mathbf{Y}^c)}. \tag{82}$$

This is then substituted into Equation (81):

$$\begin{aligned} KL(\hat{Q} \parallel \hat{P}) &= \int \hat{Q} \log \frac{p(\mathbf{Y}^r, \mathbf{Y}^c)\hat{Q}}{p(\mathbf{F}^r, \mathbf{F}^c, \mathbf{U}^r, \mathbf{U}^c, \mathbf{X}, \mathbf{Y}^r, \mathbf{Y}^c)} \\ &\quad \times d\mathbf{F}^r\, d\mathbf{F}^c\, d\mathbf{U}^r\, d\mathbf{U}^c\, d\mathbf{X} \\ &= \log p(\mathbf{Y}^r, \mathbf{Y}^c) \\ &\quad + \int \hat{Q} \log \frac{\hat{Q}}{p(\mathbf{F}^r, \mathbf{F}^c, \mathbf{U}^r, \mathbf{U}^c, \mathbf{X}, \mathbf{Y}^r, \mathbf{Y}^c)} \\ &\quad \times d\mathbf{F}^r\, d\mathbf{F}^c\, d\mathbf{U}^r\, d\mathbf{U}^c\, d\mathbf{X} \\ &\triangleq \log p(\mathbf{Y}^r, \mathbf{Y}^c) + \mathrm{A} \end{aligned} \tag{83}$$

The focus is on the integral term. Equations (18) and (23) are utilized to factorize the joint distribution and the approximated posterior, respectively:

$$\begin{aligned} \mathrm{A} &= \int \hat{Q} \log \frac{\hat{Q}}{p(\mathbf{F}^r, \mathbf{F}^c, \mathbf{U}^r, \mathbf{U}^c, \mathbf{X}, \mathbf{Y}^r, \mathbf{Y}^c)} \\ &\quad \times d\mathbf{F}^r\, d\mathbf{F}^c\, d\mathbf{U}^r\, d\mathbf{U}^c\, d\mathbf{X} \\ &= \int \hat{Q} \log \frac{q_\phi(\mathbf{X}) q_\lambda(\mathbf{U}^c) q_\gamma(\mathbf{U}^r)}{p(\mathbf{Y}^c \mid \mathbf{F}^c) p(\mathbf{U}^c) p(\mathbf{Y}^r \mid \mathbf{F}^r) p(\mathbf{U}^r) p(\mathbf{X})} \\ &\quad \times d\mathbf{F}^r\, d\mathbf{F}^c\, d\mathbf{U}^r\, d\mathbf{U}^c\, d\mathbf{X} \\ &= E_{\hat{Q}}\left[\log \frac{q_\phi(\mathbf{X}) q_\lambda(\mathbf{U}^c) q_\gamma(\mathbf{U}^r)}{p(\mathbf{Y}^c \mid \mathbf{F}^c) p(\mathbf{U}^c) p(\mathbf{Y}^r \mid \mathbf{F}^r) p(\mathbf{U}^r) p(\mathbf{X})} \right] \\ &\triangleq E_{\hat{Q}}\left[\log \frac{\tilde{Q}}{\tilde{P}} \right] \end{aligned} \tag{84}$$

Given that the KL divergence is always non-negative, the following is obtained:

$$\log p(\mathbf{Y}^r, \mathbf{Y}^c) \geq E_{\hat{Q}}\left[\log \frac{\tilde{Q}}{\tilde{P}} \right] \triangleq ELBO \tag{85}$$

Thus, a lower bound for the marginal likelihood, known as the Evidence Lower Bound (ELBO), is identified. An alternative approach to achieve the same result involves marginalizing the joint distribution, followed by multiplying and dividing the joint distribution by the joint posterior approximation:

$$\begin{aligned} p(\mathbf{Y}^r, \mathbf{Y}^c) &= \int p(\mathbf{Y}^r, \mathbf{Y}^c, \mathbf{F}^r, \mathbf{F}^c, \mathbf{U}^r, \mathbf{U}^c, \mathbf{X}) \\ &\quad \times d\mathbf{F}^r\, d\mathbf{F}^c\, d\mathbf{U}^r\, d\mathbf{U}^c\, d\mathbf{X} \\ &= \int \frac{\hat{Q}}{\tilde{Q}} p(\mathbf{Y}^r, \mathbf{Y}^c, \mathbf{F}^r, \mathbf{F}^c, \mathbf{U}^r, \mathbf{U}^c, \mathbf{X}) \\ &\quad \times d\mathbf{F}^r\, d\mathbf{F}^c\, d\mathbf{U}^r\, d\mathbf{U}^c\, d\mathbf{X} \\ &= \int \hat{Q} \frac{p(\mathbf{Y}^c \mid \mathbf{F}^c) p(\mathbf{U}^c) p(\mathbf{Y}^r \mid \mathbf{F}^r) p(\mathbf{U}^r) p(\mathbf{X})}{q_\phi(\mathbf{X}) q_\lambda(\mathbf{U}^c) q_\gamma(\mathbf{U}^r)} \\ &\quad \times d\mathbf{F}^r\, d\mathbf{F}^c\, d\mathbf{U}^r\, d\mathbf{U}^c\, d\mathbf{X} \\ &= E_{\hat{Q}}\left[\frac{p(\mathbf{Y}^c \mid \mathbf{F}^c) p(\mathbf{U}^c) p(\mathbf{Y}^r \mid \mathbf{F}^r) p(\mathbf{U}^r) p(\mathbf{X})}{q_\phi(\mathbf{X}) q_\lambda(\mathbf{U}^c) q_\gamma(\mathbf{U}^r)} \right] \\ &\triangleq E_{\hat{Q}}\left[\log \frac{\tilde{Q}}{\tilde{P}} \right] \end{aligned} \tag{86}$$

Using Jensen's inequality, the evidence lower bound (ELBO) can be found:

$$\log p(\mathbf{Y}^r, \mathbf{Y}^c) \geq E_{\hat{Q}}\left[\log \frac{\tilde{Q}}{\tilde{P}} \right] \triangleq ELBO \tag{87}$$

Having calculated the ELBO through two approaches, the next step is to decompose the ELBO into simpler terms:

$$\begin{aligned} ELBO &= E_{q(\mathbf{F}^c, \mathbf{U}^c, \mathbf{X})}\left[\log \frac{p(\mathbf{Y}^c \mid \mathbf{F}^c) p(\mathbf{U}^c)}{q_\lambda(\mathbf{U}^c)} \right] \\ &\quad + E_{q(\mathbf{F}^r, \mathbf{U}^r, \mathbf{X})}\left[\log \frac{p(\mathbf{Y}^r \mid \mathbf{F}^r) p(\mathbf{U}^r)}{q_\lambda(\mathbf{U}^c)} \right] \\ &\quad - KL(q_\phi(\mathbf{X}) \parallel p(\mathbf{X})) \\ &= E_{q_\phi(\mathbf{X})}\left[E_{p(\mathbf{F}^c \mid \mathbf{U}^c, \mathbf{X}) q_\lambda(\mathbf{U}^c)}\left[\log p(\mathbf{Y}^c \mid \mathbf{F}^c) \right] \right] \\ &\quad + E_{q_\phi(\mathbf{X})}\left[E_{p(\mathbf{F}^r \mid \mathbf{U}^r, \mathbf{X}) q_\gamma(\mathbf{U}^r)}\left[\log p(\mathbf{Y}^r \mid \mathbf{F}^r) \right] \right] \\ &\quad - KL(q_\lambda(\mathbf{U}^c) \parallel p(\mathbf{U}^c)) - KL(q_\lambda(\mathbf{U}^r) \parallel p(\mathbf{U}^r)) \\ &\quad - KL(q_\phi(\mathbf{X}) \parallel p(\mathbf{X})) \\ &= \mathrm{ELL}^{\mathrm{reg}} + \mathrm{ELL}^{\mathrm{cls}} - \mathrm{KL}_u^c - \mathrm{KL}_u^r - \mathrm{KL}_X. \end{aligned} \tag{88}$$

APPENDIX D
COMPUTATION OF PREDICTIVE DISTRIBUTION

In this appendix, the calculation of the predictive distribution is detailed. Given that the calculation for both continuous and labeled data follows the same process, the focus is on the continuous case for simplicity. The LDGD framework employs a Gaussian prior for the inducing variables, formalized as $p(\mathbf{u}^r_{:,d} \mid \mathbf{Z}) = \mathcal{N}(\mathbf{u}^r_{:,d} \mid \mathbf{0}, \mathbf{K}^r_{M_r M_r})$. Moreover, the variational distribution within this model is specified as $q_\gamma(\mathbf{U}^r) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{u}^r_{:,d} \mid \mathbf{m}^r_d, \mathbf{S}^r_d)$. Utilizing the transformation matrix A, defined as $A \triangleq \mathbf{K}^r_{NM_r} \mathbf{K}^r_{M_r M_r}{}^{-1}$, the distribution of function values, conditional on the inducing points and latent variables, is then expressed as:

$$p(\mathbf{f}^r_{:,d} \mid \mathbf{u}^r_{:,d}, \mathbf{X}) = \mathcal{N}(\mathbf{A}\mathbf{u}^r_{:,d}, \mathbf{K}^r_{NN} - \mathbf{A}\mathbf{K}^r_{M_r M_r}\mathbf{A}^T). \tag{89}$$

This formulation leverages Gaussian distribution properties for the marginal distribution. Subsequently, the predictive distribution, represented as $q_\gamma(\mathbf{f}^r_{:,d} \mid \mathbf{X})$, is derived by integrating out the inducing variables. Given that both terms within the integral are Gaussian distributions, this integration has an analytical solution, allowing for straightforward computation:

$$\begin{aligned} q_\gamma(\mathbf{f}^r_{:,d} \mid \mathbf{X}) &\triangleq \int p(\mathbf{f}^r_{:,d} \mid \mathbf{u}^r_{:,d}, \mathbf{X}) q_\gamma(\mathbf{u}^r_{:,d})\, d\mathbf{u}^r_{:,d} \\ &= \mathcal{N}(\mathbf{f}^r_{:,d} \mid \boldsymbol{\mu}^r_f, \boldsymbol{\Sigma}^r_f), \end{aligned} \tag{90}$$

$$\boldsymbol{\mu}^r_f = \mathbf{A}^T \mathbf{m}^r_d, \tag{91}$$

$$\boldsymbol{\Sigma}^r_f = \mathbf{K}^r_{NN} + \mathbf{A}^T(\mathbf{S}^r_d - \mathbf{K}^r_{M_r M_r})\mathbf{A}. \tag{92}$$

For stabilizing the learning of variational parameters, the whitening trick proposed by [55] is utilized. Let $\mathbf{L}^r$ denote the Cholesky factor of the covariance matrix of the prior over

Algorithm 2 Calculate Predictive Distribution

```
Require:
    Sampled latent variables $\mathbf{X}_b$
    Kernel matrices $\mathbf{K}_{NN}$, $\mathbf{K}_{MM}$, $\mathbf{K}_{MN}$
    Whitened variational mean $\hat{\mathbf{m}}$
    Whitened variational covariance $\hat{\mathbf{S}} = \hat{\mathbf{W}}\hat{\mathbf{W}}^T$ ($\hat{\mathbf{W}}$ is upper triangular)
1: function PredictiveDist($\mathbf{X}_b$, $\mathbf{K}_{NN}$, $\mathbf{K}_{MM}$, $\mathbf{K}_{MN}$, $\hat{\mathbf{m}}$, $\hat{\mathbf{S}}$)
2:     Add jitters to $\mathbf{K}_{MM}$
3:     $LL^T \leftarrow$ CholeskyDecomposition($\mathbf{K}_{MM}$)
4:     $\hat{\mathbf{A}} \leftarrow L^{-1}\mathbf{K}_{MN}$
5:     $\mu_f \leftarrow \hat{\mathbf{A}}^T\hat{\mathbf{m}}$
6:     $\Sigma_f \leftarrow \hat{\mathbf{A}}^T(\hat{\mathbf{W}}\hat{\mathbf{W}}^T - I)\hat{\mathbf{A}}$
7:     $\Sigma_f \leftarrow \mathbf{K}^r_{NN} + \Sigma_f$
8:     return $\mu_f$ and $\Sigma_f$
9: end function
```

Algorithm 3 Calculate Expected Log-Likelihood (ELL) for Regression

```
Require:
    Observations $\mathbf{y}$
    Predictive distribution mean and covariance $\boldsymbol{\mu}_{\mathbf{f}^r}$, $\boldsymbol{\Sigma}_{\mathbf{f}^r}$
    Noise variance $\boldsymbol{\sigma}^2$
    Sampled points $\mathbf{X}^{(j)}, j = 1, \ldots, J$
1: function ELLRegression($\mathbf{Y}^r$, $\boldsymbol{\mu}_{\mathbf{f}^r}$, $\boldsymbol{\Sigma}_{\mathbf{f}^r}$, $\boldsymbol{\sigma}^2$, $\mathbf{X}$)
2:     $ELL^{\text{reg}} \leftarrow 0$
3:     for $i = 1$ to $B$ do
4:         for $d = 1$ to $D$ do
5:             $ELL^{\text{reg}}_{i,d} \leftarrow 0$
6:             for $j = 1$ to $J$ do
7:                 $A \leftarrow \frac{(y^r_{i,d} - \mu_{\mathbf{f}^r_{:,d}}(x^{(j)}_i))^2 + \Sigma_{\mathbf{f}^r_{:,d}}(x^{(j)}_i)}{\sigma^2_d}$
8:                 $A \leftarrow A + \log 2\pi + \log \sigma^2_d$
9:                 $A \leftarrow -0.5\, A$
10:                $ELL^{\text{reg}}_{i,d} \leftarrow ELL^{\text{reg}}_{i,d} + A$
11:            end for
12:            $ELL^{\text{reg}}_{i,d} \leftarrow -\frac{1}{2J} \times ELL^{\text{reg}}_{i,d}$
13:            $ELL^{\text{reg}} \leftarrow ELL^{\text{reg}} + ELL^{\text{reg}}_{i,d}$
14:        end for
15:    end for
16:    return $ELL^{\text{reg}}$
17: end function
```

inducing variables, such that $\mathbf{K}^r_{M_rM_r} = \mathbf{L}^r(\mathbf{L}^r)^T$. Whitened parameters $\mathbf{m}^r_d = \mathbf{L}^r\hat{\mathbf{m}}^r_d$ and $\mathbf{W}^r_d = \mathbf{L}^r\hat{\mathbf{W}}^r_d$ are introduced, where $\hat{\mathbf{m}}^r_d$ and $\hat{\mathbf{W}}^r_d$ are the free parameters. Here, $\mathbf{W}^r_d$ and $\hat{\mathbf{W}}^r_d$ are upper triangular matrices. The covariance matrix of the variational distribution is computed as $\mathbf{S}^r_d = \mathbf{W}^r_d(\mathbf{W}^r_d)^T$. Given these definitions and transformations, the mean and covariance of the predictive distribution are expressed as $\mu^r_f = \hat{\mathbf{A}}^T\hat{\mathbf{m}}^r_d$ and $\Sigma^r_f = \mathbf{K}^r_{NN} + \hat{\mathbf{A}}^T\left(\hat{\mathbf{W}}^r_d(\hat{\mathbf{W}}^r_d)^T - \mathbb{I}\right)\hat{\mathbf{A}}$, where $\hat{\mathbf{A}} = L^{r\,-1}\mathbf{K}^r_{M_rN}$. The introduction of whitened parameterization and the transformation of the mean vector simplify computation, ensuring numerical stability and computational efficiency. This approach applies to categorical data (labels) as well:

Algorithm 4 Calculate Expected Log Probability Using Gauss-Hermite Quadrature

```
Require:
    Observations $\mathbf{y}$
    Predictive distribution mean and covariance
    Gauss-Hermite Quadrature nodes $Loc$
    Gauss-Hermite Quadrature weights $\omega$
1: function ELLClassification($\mathbf{Y}^c$, $\boldsymbol{\mu}_{\mathbf{f}^c}$, $\boldsymbol{\Sigma}_{\mathbf{f}^c}$, $\mathbf{X}_b$)
2:     $ELL^{\text{cls}} \leftarrow 0$
3:     for $i = 1$ to $B$ do
4:         for $k = 1$ to $K$ do
5:             $y^c_{i,k} \leftarrow 2y^c_{i,k} - 1$
6:             $ELL^{\text{cls}}_{i,k} \leftarrow 0$
7:             for $j = 1$ to $J$ do
8:                 $L \leftarrow \sqrt{2\Sigma_{f^c_k}(x^{(j)}_i)} \times Loc^{(j)} +$
9:                 $L \leftarrow L + \mu_{f^c_k}(x^{(j)}_i)$
10:                $L \leftarrow \log \Phi(L \cdot y^c_{i,k})$
11:                $ELL^{\text{cls}}_{i,k} \leftarrow ELL^{\text{cls}}_{i,k} + L \times \omega^{(j)}$
12:            end for
13:            $ELL^{\text{cls}}_{i,k} \leftarrow \frac{1}{J\sqrt{\pi}} ELL^{\text{cls}}_{i,k}$
14:            $ELL^{\text{cls}} \leftarrow ELL^{\text{cls}} + ELL^{\text{cls}}_{i,k}$
15:        end for
16:    end for
17:    return $ELL^{\text{cls}}$
18: end function
```

$$q_\lambda(\mathbf{f}^c_{:,k} \mid \mathbf{X}) \triangleq \int p(\mathbf{f}^c_{:,k} \mid \mathbf{u}^c_{:,k}, \mathbf{X}) q_\lambda(\mathbf{u}^c_{:,k})\, d\mathbf{u}^c_{:,k}$$
$$= \mathcal{N}(\mathbf{f}^c_{:,k} \mid \boldsymbol{\mu}^c_f, \boldsymbol{\Sigma}^c_f), \tag{93}$$
$$\boldsymbol{\mu}^c_f = B^T\mathbf{m}^c_k, \tag{94}$$
$$\boldsymbol{\Sigma}^c_f = \mathbf{K}^c_{NN} + B^T(\mathbf{S}^c_k - \mathbf{K}^c_{M_cM_c}), B \tag{95}$$

where $B \triangleq \mathbf{K}^c_{M_cM_c}{}^{-1}\mathbf{K}^c_{M_cN}$. Using whitened parameters, it can be expressed as $\boldsymbol{\mu}^c_f = \hat{\mathbf{B}}^T\hat{\mathbf{m}}^c_k$ and $\boldsymbol{\Sigma}^c_f = \mathbf{K}^c_{NN} + \hat{\mathbf{B}}^T(\hat{\mathbf{W}}^c_k(\hat{\mathbf{W}}^c_k)^T - \mathbb{I})\hat{\mathbf{B}}$, where $\hat{\mathbf{B}} = (\mathbf{L}^c)^{-1}\mathbf{K}^c_{M_cN}$ and $\mathbf{K}^c_{M_cM_c} = \mathbf{L}^c(\mathbf{L}^c)^T$.

Implementation details for the predictive distribution are provided in Algorithm 2. The calculation process is identical for both continuous and labeled data; hence, we omit the superscripts "r" and "c" used in the main text to differentiate between the two data types.

APPENDIX E
EXPECTED LOG-LIKELIHOOD IMPLEMENTATION

The implementation details for calculating ELL$^{\text{reg}}$ based on Equation (50) and ELL$^{\text{cls}}$ based on Equation (42) are provided in Algorithms 3 and 4, respectively.

REFERENCES

[1] R. Zebari, A. Abdulazeez, D. Zeebaree, D. Zebari, and J. Saeed, "A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction," *J. Appl. Sci. Technol. Trends*, vol. 1, no. 1, pp. 56–70, May 2020.

[2] T. H. Koh, W. E. Bishop, T. Kawashima, B. B. Jeon, R. Srinivasan, Y. Mu, Z. Wei, S. J. Kuhlman, M. B. Ahrens, S. M. Chase, and B. M. Yu, "Dimensionality reduction of calcium-imaged neuronal population activity," *Nature Comput. Sci.*, vol. 3, no. 1, pp. 71–85, Dec. 2022.

[3] S. Ma and Y. Dai, "Principal component analysis based methods in bioinformatics studies," *Briefings Bioinf.*, vol. 12, no. 6, pp. 714–722, Nov. 2011.

[4] M. U. Ali, S. Ahmed, J. Ferzund, A. Mehmood, and A. Rehman, "Using PCA and factor analysis for dimensionality reduction of bio-informatics data," 2017, *arXiv:1707.07189*.

[5] X. Zhong and D. Enke, "Forecasting daily stock market return using dimensionality reduction," *Expert Syst. Appl.*, vol. 67, pp. 126–139, Jan. 2017.

[6] J. P. Cunningham and B. M. Yu, "Dimensionality reduction for large-scale neural recordings," *Nature Neurosci.*, vol. 17, no. 11, pp. 1500–1509, Nov. 2014.

[7] W. Wang, Y. Huang, Y. Wang, and L. Wang, "Generalized autoencoder: A neural network framework for dimensionality reduction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2014, pp. 496–503.

[8] C. R. Heller and S. V. David, "Targeted dimensionality reduction enables reliable estimation of neural population coding accuracy from trial-limited data," *PLoS ONE*, vol. 17, no. 7, Jul. 2022, Art. no. e0271136.

[9] R. Bro and A. K. Smilde, "Principal component analysis," *Anal. Methods*, vol. 6, no. 9, pp. 2812–2831, 2014.

[10] S. Balakrishnama and A. Ganapathiraju, "Linear discriminant analysis—A brief tutorial," *Inst. Signal Inf. Process.*, vol. 18, pp. 1–8, Mar. 1998.

[11] M. C. Cieslak, A. M. Castelfranco, V. Roncalli, P. H. Lenz, and D. K. Hartline, "t-distributed stochastic neighbor embedding (t-SNE): A tool for eco-physiological transcriptomic analysis," *Mar. Genomics*, vol. 51, Jun. 2020, Art. no. 100723.

[12] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," 2018, *arXiv:1802.03426*.

[13] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *J. Roy. Stat. Soc. B, Stat. Methodol.*, vol. 61, no. 3, pp. 611–622, 1999.

[14] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Found. Trends Mach. Learn.*, vol. 12, no. 4, pp. 307–392, 2019.

[15] N. Lawrence and A. Hyvärinen, "Probabilistic non-linear principal component analysis with Gaussian process latent variable models," *J. Mach. Learn. Res.*, vol. 6, no. 11, pp. 1783–1816, 2005.

[16] C. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 13, 2000, pp. 1–7.

[17] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 18, 2005, pp. 1–8.

[18] M. Titsias, "Variational learning of inducing variables in sparse Gaussian processes," in *Proc. 12th Int. Conf. Artif. Intell. Statist.*, 2009, pp. 567–574.

[19] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian processes for big data," 2013, *arXiv:1309.6835*.

[20] L. Cai, X. Liu, H. Ding, and F. Chen, "Human action recognition using improved sparse Gaussian process latent variable model and hidden conditional random filed," *IEEE Access*, vol. 6, pp. 20047–20057, 2018.

[21] S. Yu, K. Yu, V. Tresp, H.-P. Kriegel, and M. Wu, "Supervised probabilistic principal component analysis," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2006, pp. 464–473.

[22] G. Baudat and F. Anouar, "Generalized discriminant analysis using a kernel approach," *Neural Comput.*, vol. 12, no. 10, pp. 2385–2404, Oct. 2000.

[23] X. Gao, X. Wang, D. Tao, and X. Li, "Supervised Gaussian process latent variable model for dimensionality reduction," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 2, pp. 425–434, Apr. 2011.

[24] C. H. Ek and P. Lawrence, "Shared Gaussian process latent variable models," Ph.D. thesis, School Eng., Comput. Math., Oxford Brookes Univ., Oxford, U.K., 2009.

[25] X. Jiang, J. Gao, T. Wang, and L. Zheng, "Supervised latent linear Gaussian process latent variable model for dimensionality reduction," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 6, pp. 1620–1632, Dec. 2012.

[26] J. Zhang, Z. Zhu, and J. Zou, "Supervised Gaussian process latent variable model based on Gaussian mixture model," in *Proc. Int. Conf. Secur., Pattern Anal., Cybern. (SPAC)*, Dec. 2017, pp. 124–129.

[27] K. Kamikawa, K. Maeda, T. Ogawa, and M. Haseyama, "Feature integration through semi-supervised multimodal Gaussian process latent variable model with pseudo-labels for interest level estimation," *IEEE Access*, vol. 9, pp. 163843–163850, 2021.

[28] N. Lawrence, "Gaussian process latent variable models for visualisation of high dimensional data," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 16, 2003, pp. 1–8.

[29] M. Seeger, "Gaussian processes for machine learning," *Int. J. Neural Syst.*, vol. 14, no. 2, pp. 69–106, 2004.

[30] H. Liu, J. Cai, and Y.-S. Ong, "Remarks on multi-output Gaussian process regression," *Knowl.-Based Syst.*, vol. 144, pp. 102–121, Mar. 2018.

[31] M. Titsias and N. D. Lawrence, "Bayesian Gaussian process latent variable model," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 844–851.

[32] W. R. Gilks, S. Richardson, and D. Spiegelhalter, *Markov Chain Monte Carlo in Practice*. Boca Raton, FL, USA: CRC Press, 1995.

[33] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *J. Amer. Stat. Assoc.*, vol. 112, no. 518, pp. 859–877, Apr. 2017.

[34] H. Salimbeni and M. Deisenroth, "Doubly stochastic variational inference for deep Gaussian processes," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–12.

[35] J. Hensman, A. Matthews, and Z. Ghahramani, "Scalable variational Gaussian process classification," in *Proc. Artif. Intell. Statist.*, 2015, pp. 351–360.

[36] M. Abramowitz, I. A. Stegun, and R. H. Romer, *Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables*. Denton, TX, USA: UNT Digital Library, 1988.

[37] D. P. Kingma, "Auto-encoding variational Bayes," 2013, *arXiv: 1312.6114*.

[38] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *Proc. NIPS Workshop Autodiff*, Long Beach, CA, USA, 2017. [Online]. Available: https://openreview.net/forum?id=BJJsrmfCZ

[39] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson, "GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–11.

[40] C. M. Bishop and G. D. James, "Analysis of multiphase flows using dual-energy gamma densitometry and neural networks," *Nucl. Instrum. Methods Phys. Res. A, Accel. Spectrom. Detect. Assoc. Equip.*, vol. 327, nos. 2–3, pp. 580–593, 1993.

[41] R. A. Fisher, 1988, "Iris," *UCI Machine Learning Repository*, doi: 10.24432/C56C76.

[42] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.

[43] *Three-Phase Flow Data*. Accessed: Feb. 7, 2024. [Online]. Available: https://inverseprobability.com/3PhaseData

[44] *UCI Machine Learning Repository*. Accessed: Feb. 7, 2024. [Online]. Available: https://archive.ics.uci.edu/ml/index.php

[45] R. F. Alvear-Sandoval, J. L. Sancho-Gómez, and A. R. Figueiras-Vidal, "On improving CNNs performance: The case of MNIST," *Inf. Fusion*, vol. 52, pp. 106–109, Dec. 2019.

[46] *MATLAB Version: 9.13.0 (r2022b)*, document R2022B, 2022.

[47] A. P. Valentine and M. Sambridge, "Gaussian process models—I. A framework for probabilistic continuous inverse theory," *Geophys. J. Int.*, vol. 220, no. 3, pp. 1632–1647, Mar. 2020.

[48] C. S. Wong and W. K. Li, "On a mixture autoregressive model," *J. Roy. Stat. Soc. B, Stat. Methodol.*, vol. 62, no. 1, pp. 95–115, 2000.

[49] R. S. Fard, N. Ziaei, and A. Yousefi, "Latent dynamical model to characterize brain network-level rhythmic dynamics," in *Proc. 45th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Jul. 2023, pp. 1–5.

[50] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," 2023, *arXiv:2312.00752*.

[51] A. Yousefi, R. S. Fard, U. T. Eden, and E. N. Brown, "State-space global coherence to estimate the spatio-temporal dynamics of the coordinated brain activity," in *Proc. 41st Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Jul. 2019, pp. 5794–5798.

[52] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural Comput.*, vol. 31, no. 7, pp. 1235–1270, Jul. 2019.

[53] R. Dey and F. M. Salem, "Gate-variants of gated recurrent unit (GRU) neural networks," in *Proc. IEEE 60th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2017, pp. 1597–1600.

[54] N. Ziaei, R. Saadatifard, A. Yousefi, B. Nazari, S. S. Cash, and A. C. Paulk, "Bayesian time-series classifier for decoding simple visual stimuli from intracranial neural activity," in *Proc. Int. Conf. Brain Informat.* Cham, Switzerland: Springer, 2023, pp. 227–238.

[55] I. Murray and R. P. Adams, "Slice sampling covariance hyperparameters of latent Gaussian models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 23, 2010, pp. 1–9.

NAVID ZIAEI received the Bachelor of Science degree in electrical, electronics, and communications engineering and the Master of Science degree in communication systems from Isfahan University of Technology, in 2016 and 2018, respectively, where he is currently pursuing the Ph.D. degree.

His work experience spans several significant roles, including his current position as a Visiting Ph.D. Student with Worcester Polytechnic Institute, USA, since April 2023. His professional endeavors are characterized by a profound engagement with neuroscience, where he specializes in the development and application of machine learning and deep learning tools. His expertise also extends to image and signal processing and statistical machine learning, with a particular emphasis on neuroscientific applications. As a Researcher, he has contributed to the field of neuroscience through innovative approaches in machine learning, aiming to unravel complex brain dynamics, and improve neurological diagnostics and treatments.

BEHZAD NAZARI received the B.Sc. degree in electrical engineering, the M.Sc. degree in biomedical engineering, and the Ph.D. degree in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 1993, 1995, and 2004, respectively.

Over the years, he has accumulated extensive experience in both industry and academia, working with companies, such as Signal Company and Royan Pardaz Company. In September 2008, he joined as a Faculty Member of Isfahan University of Technology, Isfahan, Iran, where he is currently an Assistant Professor with the Department of Electrical and Computer Engineering. His academic and research career is complemented by his contributions to various journals and conferences, enhancing his profile as a Researcher and an Educator. His research interests include signal processing, image processing, and neural computing applications. He focuses on developing computational methods for analyzing high-dimensional data and time series from neural activities, with a particular emphasis on applications in medical imaging and diagnostics.

URI T. EDEN received the B.S. degree in mathematics and in engineering and applied sciences from California Institute of Technology, in 1999, and the M.S. and Ph.D. degrees in engineering sciences from Harvard University, in 2002 and 2005, respectively.

He was a Postdoctoral Fellow with the Brain and Cognitive Sciences Department, Massachusetts Institute of Technology, from 2005 to 2006. In 2006, he joined the Department of Mathematics and Statistics, Boston University, where he is currently an Associate Professor and an Associate Director of the Program in Statistics. His research interests include developing mathematical and statistical methods to analyze neural spiking activity, integrating methodologies related to model identification, statistical inference, signal processing, and stochastic estimation and control. He received an NSF Career Award, in 2007.

ALIK S. WIDGE received the B.A. degree in computer science and cognitive science from the Dartmouth College, Hanover, NH, USA, in 1999, earning the Kemeny Computing Prize, the Ph.D. degree in robotics from Carnegie Mellon University, in 2007, focusing on innovative electrode technologies for neuro-robotic interfaces, and the M.D. degree from the School of Medicine, University of Pittsburgh, in 2009.

During the Ph.D. degree, he was recognized with the Ruth L. Kirschstein National Research Service Award. He was elected to Alpha Omega Alpha with the University of Pittsburgh. He is currently an Assistant Professor with the Department of Psychiatry, University of Minnesota Medical School, Minneapolis, MN, USA. Clinically, he specializes in brain stimulation treatments for mood, anxiety, and substance disorders with the University of Minnesota Health Park Place East Mental Health Neuromodulation Clinic. His research advances "closed-loop" devices that adapt to real-time brain signals for individualized treatment, conducted through the Translational NeuroEngineering Laboratory.

Dr. Widge is a member of several professional societies and has been honored with awards like the Charles Claude Guthrie Award for Excellence in the Basic Medical Sciences. His ongoing contributions to neuroengineering and psychiatric treatment are recognized both nationally and internationally.

ALI YOUSEFI received the M.S. degree in electrical engineering from the Sharif University of Technology, Tehran, in 2000, and the Ph.D. degree in electrical and computer engineering from the University of Southern California, Los Angeles, CA, USA, in 2014.

He completed postdoctoral training in computational neuroscience with Mayo Clinic, in 2015, followed by further postdoctoral fellowships in statistical neuroscience with the Harvard Medical School, Boston, MA, USA, and Boston University, Boston, in 2019. He is also an Associate Professor with the Department of Biomedical Engineering, University of Huston, Houston, TX, USA. He has developed numerous methodological solutions for analyzing neuroscience data. His approach is twofold: methodologically, focusing on creating statistical frameworks to link neural activity with biological and behavioral signals, developing algorithms for statistical estimation and inference, goodness-of-fit analyses, and mathematical theories applicable across various neural data modalities. Application-wise, he applies these methods to real neural data to model neural activities dynamically, characterize neural ensemble behaviors, and reproduce biological and behavioral signals in real time. His work integrates techniques in model identification, statistical inference, signal processing, Bayesian analysis, and stochastic estimation and control, enhancing the applicability of these methodologies to neural analysis models tailored for dynamics in neural systems observed through data, such as local field potentials and spike train data.