Fully Bayesian Autoencoders with Latent Sparse Gaussian Processes

Ba-Hien Tran [1] **Babak Shahbaba** [2] **Stephan Mandt** [2] **Maurizio Filippone** [1]

Abstract

We present a fully Bayesian autoencoder model that treats both local latent variables and global decoder parameters in a Bayesian fashion. This approach allows for flexible priors and posterior approximations while keeping the inference costs low. To achieve this, we introduce an amortized MCMC approach by utilizing an implicit stochastic network to learn sampling from the posterior over local latent variables. Furthermore, we extend the model by incorporating a Sparse Gaussian Process prior over the latent space, allowing for a fully Bayesian treatment of inducing points and kernel hyperparameters and leading to improved scalability. Additionally, we enable Deep Gaussian Process priors on the latent space and the handling of missing data. We evaluate our model on a range of experiments focusing on dynamic representation learning and generative modeling, demonstrating the strong performance of our approach in comparison to existing methods that combine Gaussian Processes and autoencoders.

1. Introduction

The problem of learning representations of data that are useful for downstream tasks is a crucial factor in the success of many machine learning applications (Bengio et al., 2013). Among the numerous proposed solutions, modeling approaches that evolved from autoencoders (AEs) (Cottrell et al., 1989) are particularly appealing, as they do not require annotated data and have proven effective in unsupervised learning tasks, such as data compression and generative modeling (Tomczak, 2022; Yang et al., 2022). AEs are neural networks consisting of an encoder that maps input data to a set of lower-dimensional latent codes and a decoder that maps the latent codes back to the observations.

In applications where data is scarce or uncertainty quantification is crucial, it is beneficial to treat these models in a Bayesian manner (Mackay, 1992; Neal, 1996; Wilson & Izmailov, 2020; Izmailov et al., 2021) by imposing meaningful prior distributions over both the parameters of the encoder and decoder (Tran et al., 2021; Miani et al., 2022). A parallel development are Variational Autoencoders (VAEs) (Kingma & Welling, 2014; Rezende & Mohamed, 2015) that treat the latent space of an autoencoder in a Bayesian fashion, enabling scalable inference over a large number of local (per-datapoint) latent variables using amortized variational inference. Note that these models typically treat the encoder and decoder as deterministic neural networks. In the related works section, we further elaborate on the differences between several versions of AE models and the way Bayesian inference is carried out.

A critical limitation of standard VAEs is their utilization of factorized priors, which is inadequate for modeling correlations between latent encodings. However, capturing latent correlations is often necessary to model structured data. For example, in autonomous driving or medical imaging applications, high-dimensional images are correlated in time. Spatio-temporal dependencies between samples are also common in environmental and life sciences datasets. To address this limitation, several works (Pearce, 2019; Casale et al., 2018) have attempted to introduce Gaussian process (GP) priors over the latent space of VAEs that capture correlations between pairs of latent variables through a kernel function. While GP priors outperform conventional priors on many tasks, they also introduce computational challenges such as $\mathcal{O}(N^3)$ complexity for GP inference, where N is the number of data instances. Recently, Jazbec et al. (2021) proposed the SVGP-VAE model to tackle this computational issue by relying on sparse approximations; these summarize the dataset into a set of so-called inducing points (Quiñonero-Candela & Rasmussen, 2005).

Although the SVGP-VAE model (Jazbec et al., 2021) has achieved promising results, it has several significant drawbacks. First, similarly to VAEs, SVGP-VAE is strongly tied to a variational inference (VI) formulation (Jordan et al., 1999; Zhang et al., 2018), which can lead to poor approxi-

[1]Department of Data Science, EURECOM, France [2]Departments of Statistics and Computer Science, University of California, Irvine, USA. Correspondence to: Ba-Hien Tran <ba-hien.tran@eurecom.fr>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

Table 1: A summary of related methods. Here, $\boldsymbol{\theta}, \mathbf{u}, \mathbf{S}$ refer to Gaussian Process (GP) hyperparameters, inducing variables and inducing inputs, respectively. N and B are the number of data points and the mini-batch size, whereas $M, H \ll N$ are the number of inducing points and the low-rank matrix factor, respectively. The notation ✗ indicates the nonexistence of a specific feature (column) within a given model (row), whereas the symbol - denotes that the model is not employed with a GP prior. The colors denoting the methods shall be used consistently throughout the paper.

Model	Scalable (Minibatching)	Non i.i.d. data	Free-form posterior	GP complexity	Arbitrary kernel & data type	Learnable GP	Inference $\boldsymbol{\theta}, \mathbf{u}, \mathbf{S}$	Reference
VAE	✓	✗	✗	-	-	-	-	Kingma & Welling (2014)
CVAE	✓	✗	✗	-	-	-	-	Sohn et al. (2015)
GPPVAE	✓	✓	✗	$\mathcal{O}(NH^2)$	✗	✗	✗	Casale et al. (2018)
GPVAE	✗	✓	✗	$\mathcal{O}(N^3)$	✓	✗	✗	Pearce (2019)
GP-VAE	✓	✓	✗	$\mathcal{O}(N)$	✗	✗	✗	Fortuin et al. (2020)
SGP-VAE	✓	✓	✗	$\mathcal{O}(BM^2 + M^3)$	✓	✓	✗	Ashman et al. (2020)
SVGP-VAE	✓	✓	✗	$\mathcal{O}(BM^2 + M^3)$	✓	✓	✗	Jazbec et al. (2021)
BAE	✓	✗	✓	-	-	-	-	**This work**
GP-BAE	✗	✓	✓	$\mathcal{O}(N^3)$	✓	✗	✗	**This work**
SGP-BAE	✓	✓	✓	$\mathcal{O}(BM^2 + M^3)$	✓	✓	✓	**This work**

mations due to VI making strong assumptions on both the factorization and functional form of the posterior. Second, the SVGP-VAE model follows the common practice in the sparse GP literature of optimizing the inducing points and kernel hyperparameters based on the marginal likelihood. However, this approach does not account for uncertainty in the inducing inputs and hyperparameters. It is well known that this can result in biased estimates and underestimated predictive uncertainties (Rossi et al., 2021; Lalchand et al., 2022a). In this work, we propose a novel Sparse Gaussian Process Bayesian Autoencoder (SGP-BAE) model that addresses these issues by providing scalable and flexible inference through a fully Bayesian treatment without relying on VI.

Contributions. Specifically, **first**, we develop a fully Bayesian autoencoder (BAE) model (§ 3), where we adopt a Bayesian treatment for both the local (per-datapoint) latent variables and the global decoder parameters. This approach differs from VAEs in that it allows specifying any prior over the latent space while decoupling the model from the inference. As a result, we can rely on powerful alternatives to VI to carry out inference, and we adopt stochastic gradient Hamiltonian Monte Carlo (SGHMC) (Chen et al., 2014) as a scalable solution. To achieve this, we propose an amortized Markov chain Monte Carlo (MCMC) approach for our Bayesian autoencoder (BAE) model by using an implicit stochastic network as the encoder to learn to draw samples from the posterior of local latent variables. Our approach addresses the prohibitively expensive inference cost induced by the local latent variables and avoids making strong assumptions on the form of the posterior. **Second**, when imposing a GP prior over the latent space, we propose a novel scalable SGP-BAE model (§ 4) in which the inducing points and kernel hyperparameters of the sparse GP prior on the latent space are treated in a fully Bayesian manner. This model offers attractive features such as high scalability, richer modeling capability, and improved prediction quality. **Third**, we extend the SGP-BAE model to allow one to impose deep GP priors on the latent space (Damianou & Lawrence, 2013) and to handle missing data. To the best of our knowledge, this is the first work to consider deep GP priors for AEs. **Finally**, we conduct a rigorous evaluation of our SGP-BAE model through a variety of experiments on dynamic representation learning and generative modeling. The results demonstrate excellent performance compared to existing methods of combining GPs and AEs (§ 5).

2. Related Work

Autoencoders. AEs (Cottrell et al., 1989) are powerful models for representation learning which operate by projecting data onto a lower-dimensional latent space through an encoder and by mapping latent representations back into the original data by means of a decoder. VAEs (Kingma & Welling, 2014; Rezende et al., 2014) elegantly combine AEs with variational inference enabling the model to generate new data and allowing for the specification of any prior on the latent space. To improve the performance of VAEs, recent works have attempted to employ flexible priors such as mixtures (Dilokthanakul et al., 2016; Tomczak & Welling, 2018; Bauer & Mnih, 2019), normalizing flows (Chen et al., 2017), nonparametric models such as Stick-breaking processes (Nalisnick & Smyth, 2017), or Gaussian processes (Casale et al., 2018).

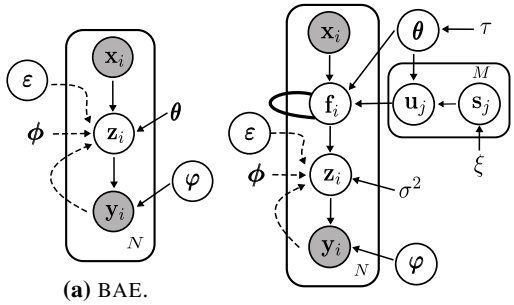
Recently, Tran et al. (2021) and Miani et al. (2022) explored a Bayesian treatment of the encoder and decoder parameters in standard AEs, demonstrating superior performance. However, this approach lacks a mechanism to impose priors on the latent space. In the seminal work on VAE, Kingma & Welling (2014) already explored a fully Bayesian treatment of VAEs by introducing a prior on the decoder. Variational

inference is employed to infer the decoder and the latent variables. Daxberger & Hernández-Lobato (2019) suggested employing SGHMCs for sampling decoder parameters, but this method uses the evidence lower bound (ELBO) of VAEs as the sampling objective, which may lead to suboptimal approximations. Following Glazunov & Zarras (2022) we use the term Bayesian Variational Autoencoder (BVAE) to refer to this set of models. To avoid confusion with these models, hereafter, we use the term Bayesian autoencoders (BAEs) to indicate our proposed approach, where both the latent variables and decoder are treated in a fully Bayesian way, and inference uses our amortized SGHMC scheme.

Gaussian process priors for AE models. The earliest attempts to combine AE models with GPs are the GP prior VAE (GPPVAE) (Casale et al., 2018) and GP-VAE (Pearce, 2019). Both these models lack scalability for generic kernel choices and data types. GPPVAE is restricted to a specialized GP product kernel and employs a Taylor approximation for GPs, while GP-VAE relies on exact GP inference. Recently, Fortuin et al. (2020) and Zhu et al. (2022) propose GP-VAE and Markovian-GPVAE, respectively, that are indeed scalable (linear in N time complexity) by exploiting the Markov assumption, but they are applicable only on time-series data. Most closely to our method is the approach of Jazbec et al. (2021) (SVGP-VAE), Ashman et al. (2020) (SGP-VAE) and Ramchandran et al. (2021), where they utilize inducing point methods (Titsias, 2009; Hensman et al., 2013) to make GPs scalable. However, all these methods strongly rely on VAEs and a variational formulation for GPs. In this work, we take a completely different route, as we aim to treat sparse GPs and AEs in a fully Bayesian way while enjoying scalability thanks to recent advances in stochastic-gradient MCMC sampling. Table 1 compares our proposed models with relevant related works.

3. Imposing Distributions over the Latent Space of Bayesian Autoencoders

We are interested in unsupervised learning problems with a high-dimensional dataset consisting of N data points $\mathbf{Y} \stackrel{\text{def}}{=} [\mathbf{y}_1, \cdots, \mathbf{y}_N]^\top \in \mathbb{R}^{N \times P}$. Each data point has a corresponding low-dimensional auxiliary data entry, summarized as $\mathbf{X} \stackrel{\text{def}}{=} [\mathbf{x}_1, \cdots, \mathbf{x}_N]^\top \in \mathbb{R}^{N \times D}$. For instance, $\mathbf{y}_i$ could be a video frame and $\mathbf{x}_i$ the corresponding time stamp. As another example, consider electronic health record (EHR) data, where the auxiliary data could include patient covariate information, such as age, height, weight, sex, and time since remission. Finally, we denote by $\mathbf{Z} \stackrel{\text{def}}{=} [\mathbf{z}_1, \cdots, \mathbf{z}_N]^\top \in \mathbb{R}^{N \times C}$ the low-dimensional latent representation of the data, meaning that each latent variable $\mathbf{z}_i$ lives in a C-dimensional latent space. We aim to build a model that is able to (1) generate $\mathbf{Y}$ based on the auxiliary data $\mathbf{X}$, and (2) provide useful and interpretable low-dimensional representations of $\mathbf{Y}$.



(a) BAE. **(b)** SGP-BAE.

Figure 1: The graphical models of vanilla BAE **(a)**, and the proposed SGP-BAE with a fully Bayesian sparse GP prior imposed on the latent space. Here, we treat the decoder's parameters $\boldsymbol{\varphi}$ in a Bayesian way while the encoder's parameters $\boldsymbol{\phi}$ are considered deterministically. The solid lines denote the generative part, whereas the dashed lines denote the encoding part. The SGP-BAE is treated in a fully Bayesian manner by imposing priors on the decoder's parameters $\boldsymbol{\varphi}$ as well as the GP kernel's parameters $\boldsymbol{\theta}$, inducing locations $\mathbf{s}$. The cyclic thick line represents that the latent GP correlates with every latent code.

Model setup. In this work, we consider a model based on AEs, and we aim to treat this in a fully Bayesian manner. This treatment promises improved predictions, reliable uncertainty estimates, and increased robustness under data sparsity (Mackay, 1992; Izmailov et al., 2021). One difficulty in doing so is that the prior distribution over the latent variables would be determined by the prior over the weights of the encoder and not a distribution of interest. In many applications, including the ones considered here, this is undesirable, and the goal is to impose a certain prior distribution over the latent representation in a similar vein as VAEs and their variants. Therefore, we propose to treat the entire AE in a fully Bayesian manner except for the encoder and to design the encoder in such a way that it maps data $\mathbf{y}_i$ into corresponding codes $\mathbf{z}_i$ while allowing these mappings to be compatible with posterior samples over the latent codes. For the encoder, as we will elaborate on shortly, we will employ a so-called stochastic inference network to learn to draw posterior samples of latent variables $\mathbf{z}_i$ given high-dimensional inputs $\mathbf{y}_i$, while for the decoder and the latent variables we employ scalable MCMC techniques.

Bayesian treatment of the latent space and the decoder. In order to retain a fully Bayesian treatment of the latent space and the decoder, we impose a prior $p(\boldsymbol{\varphi})$ over the decoder's parameters, $\boldsymbol{\varphi}$. In addition, another prior $p(\mathbf{Z} \mid \mathbf{X}; \boldsymbol{\theta})$ is imposed on the latent variables. This prior is conditioned on the auxiliary data $\mathbf{X}$ and characterized by a set of hyperparameters $\boldsymbol{\theta}$. For example, one may employ an uninformative prior such as an isotropic Gaussian commonly used in standard VAEs, but in the next section we will consider

structured priors such as GPs and their deep version as well. We assume that the observed data $\mathbf{Y}$ is fully factorized and conditional on the latent variables $\mathbf{Z}$ and a decoder network with parameters $\boldsymbol{\varphi}$, i.e., $p(\mathbf{Y} \mid \mathbf{Z}, \boldsymbol{\varphi}) = \prod_{i=1}^{N} p(\mathbf{y}_i \mid \mathbf{z}_i, \boldsymbol{\varphi})$. The full joint distribution of the model is as follows:

$$p(\boldsymbol{\varphi}, \mathbf{Z}, \mathbf{Y} \mid \mathbf{X}) = p(\boldsymbol{\varphi}) p(\mathbf{Z} \mid \mathbf{X}; \boldsymbol{\theta}) p(\mathbf{Y} \mid \mathbf{Z}, \boldsymbol{\varphi}). \quad (1)$$

We wish to infer the posterior over the latent variables and decoder parameters, which is given by Bayes' rule:

$$p(\boldsymbol{\varphi}, \mathbf{Z} \mid \mathbf{Y}, \mathbf{X}) = \frac{p(\boldsymbol{\varphi}, \mathbf{Z}, \mathbf{Y} \mid \mathbf{X})}{p(\mathbf{Y} \mid \mathbf{X})}, \quad (2)$$

where $p(\mathbf{Y} \mid \mathbf{X}) = \int p(\boldsymbol{\varphi}, \mathbf{Z}, \mathbf{Y} \mid \mathbf{X}) d\boldsymbol{\varphi} d\mathbf{Z}$ is the marginal likelihood. The generative process of data samples from this BAE model is illustrated in Fig. 1a.

Characterizing the posterior distribution over $\boldsymbol{\varphi}, \mathbf{Z}$ is analytically intractable and requires approximations. Given the success of scalable MCMC techniques to obtain samples from the posterior of model parameters in deep learning models (Zhang et al., 2020; Tran et al., 2022), in this work, we propose to follow this practice to obtain samples from $p(\boldsymbol{\varphi}, \mathbf{Z} \mid \mathbf{Y}, \mathbf{X})$. In particular, we employ SGHMC (Chen et al., 2014), which can scale up to large datasets by relying on noisy unbiased estimates of the energy function (log-posterior) $U(\boldsymbol{\varphi}, \mathbf{Z}; \mathbf{X}, \mathbf{Y}) \stackrel{\text{def}}{=} -\log p(\boldsymbol{\varphi}, \mathbf{Z}, \mathbf{Y} \mid \mathbf{X})$ and without the need to evaluate the energy function over the entire data set. More precisely, when the prior over the latent codes is fully factorized, we can approximate this energy function using mini-batches of size B as follows:

$$U(\boldsymbol{\varphi}, \mathbf{Z}; \mathbf{X}, \mathbf{Y}) \approx \tilde{U}(\boldsymbol{\varphi}, \mathbf{Z}; \mathbf{X}, \mathbf{Y}) = -\log p(\boldsymbol{\varphi}) - \quad (3)$$
$$-\frac{N}{B} \sum_{i \in \mathcal{I}_B} \left[\log p(\mathbf{z}_i \mid \mathbf{x}_i; \boldsymbol{\theta}) + \log p(\mathbf{y}_i \mid \mathbf{z}_i, \boldsymbol{\varphi}) \right]$$

where $\mathcal{I}_B$ is a set of B random indices. The exact procedure for generating samples from the posterior over $\boldsymbol{\varphi}, \mathbf{Z}$ using SGHMC can be found in Appendix A.

Encoder as a stochastic inference network. SGHMC can be challenging to implement on probabilistic models with many latent variables due to the high computational burden of iteratively refining the approximate posterior for each latent variable. Additionally, it can be difficult to evolve the latent variables for each new test sample. To address these challenges, we propose using a stochastic neural network as an inference network to efficiently generate latent codes similar to those generated by the posterior distribution, inspired by amortized inference techniques (Kingma & Welling, 2014; Wang & Liu, 2016; Feng et al., 2017; Shi et al., 2019) and MCMC distillation (Korattikara Balan et al., 2015; Wang et al., 2018; Li et al., 2017).

More specifically, instead of storing every latent code, we use an inference network $\mathbf{z}_i = f_{\boldsymbol{\phi}}(\mathbf{y}_i; \boldsymbol{\varepsilon})$ with parameters $\boldsymbol{\phi}$ that generates a corresponding latent code $\mathbf{z}_i$ given an input $\mathbf{y}_i$ and a random seed $\boldsymbol{\varepsilon}$. The random seed $\boldsymbol{\varepsilon}$ is drawn from a distribution $q(\boldsymbol{\varepsilon})$ that is easy to sample from, such as a uniform or standard Gaussian distribution. The inference network f_{ϕ} serves as an encoder by generating posterior samples of the latent code $\mathbf{z}_i$ given the observed input $\mathbf{y}_i$. It is essential to note that the encoder in our model approximates the posterior distribution of latent variables, which is similar to the approach used in VAEs. However, the primary distinction is that we do not assume any specific form of the posterior distribution of latent variables $\mathbf{z}_i$. Our encoder is trained to draw posterior samples of latent variables, rather than serving as a parametric variational distribution.

We incrementally refine the encoder $f_{\boldsymbol{\phi}}$ such that its outputs mimic the SGHMC dynamics. Specifically, after every K iterations of sampling the decoder parameters and the latent codes using SGHMC, we adjust the encoder parameters $\boldsymbol{\phi}$ based on the following objective:

$$\mathcal{L}\left(\{\mathbf{z}_i, \mathbf{y}_i\}_{i \in \mathcal{I}_B}; \boldsymbol{\phi}\right) \stackrel{\text{def}}{=} \sum_{i \in \mathcal{I}_B} \left\| f_{\boldsymbol{\phi}}(\mathbf{y}_i; \boldsymbol{\varepsilon}_i) - \mathbf{z}_i^{(k)} \right\|_2^2, \quad (4)$$

where $\mathbf{z}_i^{(k)}$ is the k-th posterior sample from SGHMC of the latent variable $\mathbf{z}_i$, and it is used as label to update $\boldsymbol{\phi}$. As the analytic solution of Eq. 4 is intractable, we perform J steps of gradient descent to update $\boldsymbol{\phi}$ using an optimizer such as Adam (Kingma & Ba, 2015). It is worth mentioning that our objective to train the inference network is a modeling choice. Conditional generative models such as diffusion models (Ho et al., 2020) or Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) can be considered as alternatives. However, we must exercise caution to ensure that our goal of learning low-dimensional representations of the data is met. In our experiments, we used a similar network architecture for the inference network as in VAEs for fair comparisons. The inference procedure for our BAEs is described in Algorithm 1.

4. Scalable Gaussian Process Prior for Bayesian Autoencoders

In the previous section, we introduced our novel version of a BAE where we imposed a simple fully-factorized prior over the latent space, such as isotropic Gaussians. However, in many applications, such priors are incapable of appropriately modeling the correlation nature of the data. For example, it is sensible to model structured data evolving over time with a BAE with a prior over the latent space in the form of a GP with the auxiliary data as input. In this section, we consider these scenarios precisely by introducing GP priors in the latent space, which allows us to model sample covariances as a function of the auxiliary data. We then discuss the scalability issues induced by the use of GP priors, and we propose Sparse Gaussian Process Bayesian Autoen-

Algorithm 1: Inference for BAEs with SGHMC

Input: Dataset $\{\mathbf{X}, \mathbf{Y}\}$, mini-batch size B, # SGHMC iterations K, # encoder interations J

Initialize the autoencoder parameters $\boldsymbol{\phi}$ and $\boldsymbol{\varphi}$
while $\boldsymbol{\varphi}$, $\mathbf{Z}$ *have not converged* **do**
  Sample a mini-batch of B random indices $\mathcal{I}_B$
  Sample random seed $\{\boldsymbol{\varepsilon}_i\}_{i=1}^B \sim q(\boldsymbol{\varepsilon})$
  Initialize the latent codes from the encoder $\{\mathbf{z}_i\}_{i=1}^B = f_{\boldsymbol{\phi}}(\{\mathbf{y}_i\}_{i\in\mathcal{I}_B}, \{\boldsymbol{\varepsilon}_i\}_{i=1}^B)$
  for *K iterations* **do**
    Compute energy function $\tilde{U}$ using Eq. 3
    Sample from posterior $p(\boldsymbol{\varphi}, \mathbf{Z} \mid \mathbf{Y}, \mathbf{X})$: $\boldsymbol{\varphi}, \{\mathbf{z}\}_{i=1}^B \leftarrow \texttt{SGHMC}(\boldsymbol{\varphi}, \{\mathbf{z}\}_{i=1}^B; \nabla_{\boldsymbol{\varphi},\mathbf{z}}\tilde{U})$
  for *J iterations* **do**
    Compute objective function $\mathcal{L}$ using Eq. 4
    Update encoder: $\boldsymbol{\phi} \leftarrow \texttt{Optimizer}(\boldsymbol{\phi}; \nabla_{\boldsymbol{\phi}}\mathcal{L})$

coders (SGP-BAEs) where we recover scalability thanks to sparse approximations to GPs (Quiñonero-Candela & Rasmussen, 2005; Rossi et al., 2021). In this model, we carry out fully Bayesian inference of the decoder, as well as the inducing inputs and covariance hyperparameters of the sparse GPs, while we optimize the stochastic inference network implementing the encoder.

4.1. Gaussian process prior

We assume C independent latent functions $f^{[1]}, ..., f^{[C]}$, which results in each $\mathbf{z}_i$ being evaluated at the corresponding $\mathbf{x}_i$, i.e., $\mathbf{z}_i = \left[f^{[1]}(\mathbf{x}_i), \cdots, f^{[C]}(\mathbf{x}_i)\right]$. We assume that each function is drawn from a zero-mean GP prior with a covariance function $\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta})$:

$$p(\mathbf{Z} \mid \mathbf{X}; \boldsymbol{\theta}) = \prod_{c=1}^{C} \mathcal{N}\left(\mathbf{z}_{1:N}^{[c]} \mid \mathbf{0}, \mathbf{K}_{\mathbf{xx}\mid\boldsymbol{\theta}}\right), \tag{5}$$

where the c-th latent channel of all latent variables, $\mathbf{z}_{1:N}^{[c]} \in \mathbb{R}^N$ (the c-th column of $\mathbf{Z}$), has a correlated Gaussian prior with covariance $\mathbf{K}_{\mathbf{xx}\mid\boldsymbol{\theta}} \in \mathbb{R}^{N\times N}$ obtained by evaluating $\kappa(\mathbf{x}_i, \mathbf{x}_j; \boldsymbol{\theta})$ over all input pairs of $\mathbf{X}$. Here, the latent function values are informed by all $\mathbf{y}$ values according to the covariance of the corresponding auxiliary input $\mathbf{x}$. One can recover the fully factorized $\mathcal{N}(\mathbf{0}, \mathbf{I})$ prior on the latent space by simply setting $\mathbf{K}_{\mathbf{xx}\mid\boldsymbol{\theta}} = \mathbf{I}$.

This GP prior over the latent space of BAEs introduces fundamental scalability issues. First, we have to compute the inverse and log-determinant of the kernel matrix $\mathbf{K}_{\mathbf{xx}\mid\boldsymbol{\theta}}$, which results in $\mathcal{O}(N^3)$ time complexity. This is only possible when N is of moderate size. Second, it is impossible to employ a mini-batching inference method like SGHMC since the energy function $U(\boldsymbol{\varphi}, \mathbf{Z}; \mathbf{X}, \mathbf{Y})$ does not decompose as a sum over all the observations.

4.2. Bayesian sparse Gaussian processes

In order to keep the notation uncluttered, we focus on a single channel and suppress the superscript index c. Given a set of latent function evaluations over the dataset, $\mathbf{f} = [f_1, \cdots, f_N]^\top$, we assume that the latent codes are stochastic realizations based on $\mathbf{f}$ and additive Gaussian noise, i.e., $\mathcal{N}(\mathbf{Z} \mid \mathbf{f}, \sigma^2\mathbf{I})$. Sparse GPs (Quiñonero-Candela & Rasmussen, 2005) are a family of approximate models that address the scalability problem by introducing a set of $M \ll N$ inducing points $\mathbf{u} = (u_1, \cdots, u_M)$ at corresponding inducing inputs $\mathbf{S} = \{\mathbf{s}_1, \cdots, \mathbf{s}_M\}$ such that $u_i = f(\mathbf{s}_i)$. We assume that these inducing variables follow the same GP as the original process, resulting in the following joint prior:

$$p(\mathbf{f}, \mathbf{u}) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}_{\mathbf{XX}\mid\boldsymbol{\theta}} & \mathbf{K}_{\mathbf{XS}\mid\boldsymbol{\theta}} \\ \mathbf{K}_{\mathbf{SX}\mid\boldsymbol{\theta}} & \mathbf{K}_{\mathbf{SS}\mid\boldsymbol{\theta}} \end{bmatrix}\right), \tag{6}$$

where the covariance matrices $\mathbf{K}_{\mathbf{SS}\mid\boldsymbol{\theta}}$ and $\mathbf{K}_{\mathbf{XS}\mid\boldsymbol{\theta}}$ are computed between the elements in $\mathbf{S}$ and $\{\mathbf{X}, \mathbf{S}\}$, respectively.

Fully Bayesian sparse GPs. The fully Bayesian treatment of sparse GPs requires priors $p_\tau(\boldsymbol{\theta})$ and $p_\xi(\mathbf{S})$ over covariance hyperparameters and inducing inputs, respectively, with τ and ξ as prior hyperparameters. With these assumptions, we term this model as Bayesian sparse Gaussian process autoencoder (SGP-BAE), and the corresponding generative model is illustrated in Fig. 1b.

By defining $\boldsymbol{\Psi} \stackrel{\text{def}}{=} \{\boldsymbol{\varphi}, \mathbf{u}, \mathbf{S}, \boldsymbol{\theta}\}$, we can rewrite the full joint distribution of parameters in SGP-BAE:

$$\begin{aligned} &p(\boldsymbol{\Psi}, \mathbf{f}, \mathbf{Z}, \mathbf{Y} \mid \mathbf{X}) = \\ &= \underbrace{p(\boldsymbol{\Psi})}_{\substack{\text{Priors on inducing}\\ \text{inputs \& variables, decoder}}} \underbrace{p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{S}, \boldsymbol{\theta}) p(\mathbf{Z} \mid \mathbf{f}; \sigma^2\mathbf{I})}_{\text{Sparse GP prior on latent space}} \underbrace{p(\mathbf{Y} \mid \mathbf{Z}, \boldsymbol{\varphi})}_{\substack{\text{Likelihood of}\\ \text{observed data}}}, \end{aligned} \tag{7}$$

where $p(\boldsymbol{\Psi}) = p(\boldsymbol{\varphi})p_\tau(\boldsymbol{\theta})p_\xi(\mathbf{S})p(\mathbf{u} \mid \mathbf{S}, \boldsymbol{\theta})$. Here, $p(\mathbf{u} \mid \mathbf{S}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{SS}\mid\boldsymbol{\theta}})$, and $p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{S}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{K}_{\mathbf{XS}\mid\boldsymbol{\theta}}\mathbf{K}_{\mathbf{SS}\mid\boldsymbol{\theta}}^{-1}\mathbf{u}, \mathbf{K}_{\mathbf{XX}\mid\boldsymbol{\theta}} - \mathbf{K}_{\mathbf{XS}\mid\boldsymbol{\theta}}\mathbf{K}_{\mathbf{SS}\mid\boldsymbol{\theta}}^{-1}\mathbf{K}_{\mathbf{SX}\mid\boldsymbol{\theta}})$. We assume a factorization $p(\mathbf{Z} \mid \mathbf{f}; \sigma^2\mathbf{I}) = \prod_{i=1}^N p(\mathbf{z}_i \mid f_i; \sigma^2)$ and make no further assumptions about the other distribution.

Scalable inference objective. We wish to infer the set of variables $\boldsymbol{\Psi}$ and the latent codes $\mathbf{Z}$. To do so, we have to marginalize out the latent process $\mathbf{f}$ from the full joint distribution above. In particular, we have:

$$\begin{aligned} &\log p(\boldsymbol{\Psi}, \mathbf{Z}, \mathbf{Y} \mid \mathbf{X}) = \log p(\boldsymbol{\Psi}) + \\ &+ \log \int p(\mathbf{f} \mid \boldsymbol{\Psi}, \mathbf{X}) p(\mathbf{Z} \mid \mathbf{f}, \sigma^2\mathbf{I}) d\mathbf{f} + \log p(\mathbf{Y} \mid \mathbf{Z}, \boldsymbol{\varphi}). \end{aligned} \tag{8}$$

This objective should be decomposed over observations to sample from the posterior over all the latent variables using a scalable method such as SGHMC. As discussed by Rossi et al. (2021), this can be done effectively by imposing independence in the conditional distribution (Snelson & Ghahramani, 2005), i.e., by parameterizing $p(\mathbf{f} \mid \boldsymbol{\Psi}, \mathbf{X}) = \mathcal{N}\big(\mathbf{K}_{\mathbf{xs} \mid \boldsymbol{\theta}} \mathbf{K}_{\mathbf{ss} \mid \boldsymbol{\theta}}^{-1} \mathbf{u}, \operatorname{diag}\big[\mathbf{K}_{\mathbf{xx} \mid \boldsymbol{\theta}} - \mathbf{K}_{\mathbf{xs} \mid \boldsymbol{\theta}} \mathbf{K}_{\mathbf{ss} \mid \boldsymbol{\theta}}^{-1} \mathbf{K}_{\mathbf{sx} \mid \boldsymbol{\theta}}\big]\big)$. With this approximation, the log-joint marginal becomes as follows:

$$\log p(\boldsymbol{\Psi}, \mathbf{Z}, \mathbf{Y} \mid \mathbf{X}) \approx \log p(\boldsymbol{\Psi}) + \tag{9}$$
$$+ \sum_{n=1}^{N} \Big\{ \log \mathbb{E}_{p(f_n \mid \boldsymbol{\Psi}, \mathbf{X})}[p(\mathbf{z}_n \mid f_n; \sigma^2)] + \log p(\mathbf{y}_n \mid \mathbf{z}_n, \boldsymbol{\varphi}) \Big\}$$
$$\stackrel{\text{def}}{=} -U(\boldsymbol{\Psi}, \mathbf{Z}; \mathbf{X}, \mathbf{Y}).$$

We can now carry out inference for this SGP-BAE model by plugging a mini-batching approximation of this energy function into Line 7 and Line 8 of Algorithm 1. By using this sparse approximation, we reduce the computational complexity of evaluating the GP prior down to $\mathcal{O}(BM^2)$, and SGP-BAE can be readily applied to a generic dataset and arbitrary GP kernel function.

Extension to deep Gaussian processes. We can easily extend SGP-BAE to deep Gaussian process priors (Damianou & Lawrence, 2013) to model much more complex functions in the latent space of BAEs. To the best of our knowledge, the use of deep GPs has not been considered in previous work. We assume a deep Gaussian process prior $f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)}$, where each $f^{(l)}$ is a GP. Each layer is associated with a set of kernel hyperparameters $\boldsymbol{\theta}^{(l)}$, inducing inputs $\mathbf{S}^{(l)}$ and inducing variables $\mathbf{u}^{(l)}$.The set of variables to be inferred is $\boldsymbol{\Psi} = \{\boldsymbol{\varphi}\} \cup \{\mathbf{u}^{(l)}, \mathbf{S}^{(l)}, \boldsymbol{\theta}^{(l)}\}_{l=1}^{L}$. The joint distribution is as follows:

$$p(\boldsymbol{\Psi}, \{\mathbf{f}^{(l)}\}_{l=1}^{L}, \mathbf{Z}, \mathbf{Y} \mid \mathbf{X}) = \tag{10}$$
$$= p(\boldsymbol{\Psi}) \underbrace{\prod_{l=1}^{L} p(\mathbf{f}^{(l)} \mid \mathbf{f}^{(l-1)}, \boldsymbol{\Psi}) p(\mathbf{Z} \mid \mathbf{f}^{(L)}; \sigma^2 \mathbf{I})}_{\text{Deep GP prior}} p(\mathbf{Y} \mid \mathbf{Z}, \boldsymbol{\varphi}),$$

where we omit the dependency on $\mathbf{X}$ for notational brevity. To perform inference, the hidden layers $\mathbf{f}^{(l)}$ have to be marginalized and propagated up to the final layer L (Salimbeni & Deisenroth, 2017). The marginalization can be approximated by quadrature (Hensman et al., 2015a) or through Monte Carlo sampling (Bonilla et al., 2019). Detailed derivations of this extension can be found in Appendix C.

Extension for missing data. In practice, real-world data may be sparse, with many missing and few overlapping dimensions across the entire dataset. We can easily extend SGP-BAE to handle such datasets. We assume that any possible permutation of observed features is potentially missing, such that each high-dimensional observation $\mathbf{y}_n = \mathbf{y}_n^o \cup \mathbf{y}_n^u$ contains a set of observed features $\mathbf{y}_n^o$ and unobserved features $\mathbf{y}_n^u$. The likelihood term of the inference objective (Eq. 10) factorizes across data points and dimensions, so there is no major modification in this objective, as the summation of the likelihood term should be done only over the non-missing dimensions, i.e.:

$$p(\mathbf{Y} \mid \mathbf{Z}, \boldsymbol{\varphi}) = \sum_{n=1}^{N} \log p(\mathbf{y}_n^o \mid \mathbf{z}_n, \boldsymbol{\varphi}). \tag{11}$$

5. Experiments

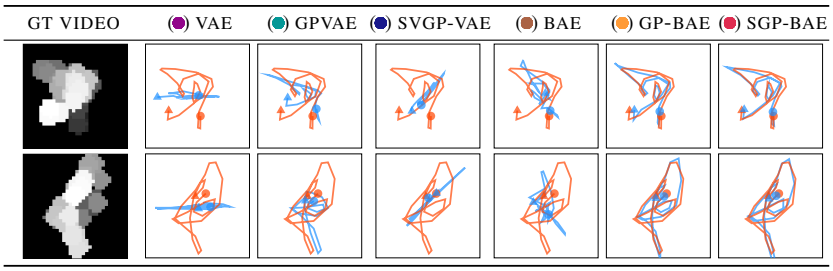
In this section, we provide empirical evidence that our SGP-BAE outperforms alternatives of combination between GP priors and AE models on synthetic data and real-world high-dimensional benchmark datasets. Throughout all experiments, unless otherwise specified, we use the radial basis function (RBF) kernel with automatic relevance determination (ARD) with marginal variance and independent lengthscales λ_i per feature (MacKay, 1996). We place a lognormal prior with unit variance and means equal to 1 and 0.05 for the lengthscales and variance, respectively. Since the auxiliary data of most of the considered datasets are timestamps, we impose a non-informative uniform prior on the inducing inputs. We observe that this prior works well in our experiments. We set the hyperparameters of the number of SGHMC and optimization steps to $J = 30$, and $K = 50$, respectively. The details for all experiments are available in Appendix D.

5.1. Synthetic moving ball data

We begin our empirical evaluation by considering the moving ball dataset proposed by Pearce (2019). This dataset comprises grayscale videos showing the movement of a ball. The two-dimensional trajectory of the ball is simulated from a GP characterized by an RBF kernel. Our task is to reconstruct the underlying trajectory in the 2D latent space from the frames in pixel space. Unlike Jazbec et al. (2021), we generate a fixed number of 35 videos for training and another 35 videos for testing. It is still possible to perform full GP inference on such a small dataset. For this experiment, we consider full GP-based methods, such as Gaussian Process Bayesian Autoencoder (GP-BAE) and GP-VAE (Pearce, 2019), as oracles for the sparse variants. Because the dataset is quite small, we perform full-batch training/inference.

Benefits of moving away from variational inference In this experiment, we show that, by relaxing strong assumptions on the posterior of latent space and taking advantage

Table 2: Reconstructions of the latent trajectories of moving ball. In the first column, frames of each test video are overlayed and shaded by time. Ground truth trajectories are illustrated in **orange**, while predicted trajectories are depicted in **blue**. We use $M = 10$ inducing points for the methods employed with sparse GPs.



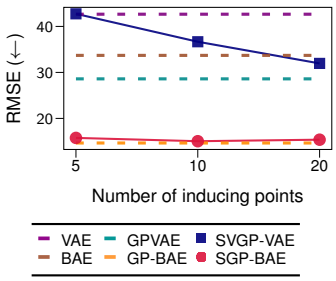


Figure 2: Performance of autoencoder models as a function of the number of inducing points.

of a powerful scalable MCMC method, BAEs consistently outperform VAEs. Fig. 2 illustrates the performance of the considered methods in terms of root mean squared error (RMSE). The results show that our GP-BAE model performs much better than GP-VAE (Pearce, 2019) though both models use the same full GP priors. In addition, by treating inducing inputs and kernel hyperparameters of sparse GPs in a Bayesian fashion, SGP-BAE offers a rich modeling capability. This is evident in the improved performance of SGP-BAE compared to sparse variational Gaussian process VAE (SVGP-VAE) (Jazbec et al., 2021). SGP-BAE is able to approach the performance of GP-BAE despite using a small number of inducing points. These numerical results align with the qualitative evaluation of the reconstructed trajectories shown in Table 2. As expected, the standard BAE and VAE endowed with a $\mathcal{N}(\mathbf{0}, \mathbf{I})$ prior on the latent space completely fail to model the trajectories faithfully. In contrast, GP-BAE and SGP-BAE are able to match them closely. In Appendix F, we further show an ablation study on alternatives of Bayesian treatments for AEs and AE-style models with GP prior. This study ultimately demonstrates that our proposal offers superior performance.

Benefits of being fully Bayesian. Our SGP-BAE model has the same advantage as the SVGP-VAE (Jazbec et al., 2021) in that it allows for an arbitrary GP kernel, and the kernel hyperparameters and inducing inputs can be inferred jointly during training or inference. In contrast, other methods either use fixed GP priors (Pearce, 2019) or employ a two-stage approach (Casale et al., 2018), where the GP hyperparameters are optimized separately from the AEs. SVGP-VAE optimizes these hyperparameters using the common practice of maximization of the marginal likelihood, ML-II. This results in a point estimate of the hyperparameters but may be prone to overfitting, especially when the training data is small while there are many hyperparameters. A distinct advantage of SGP-BAE over SVGP-VAE is that it is fully Bayesian for the GP hyperparameters and inducing points. This not only improves the quality of the predictions but also offers sensible uncertainty quantification. Fig. 3 illustrates the posterior of the lengthscale. By

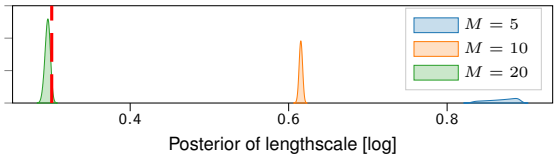


Figure 3: The posterior of the lengthscale corresponding to using a different number of inducing points, M. The **red** line denotes the true lengthscale.

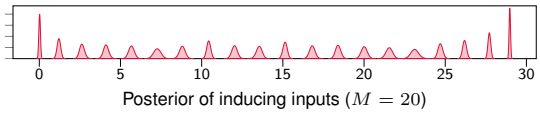


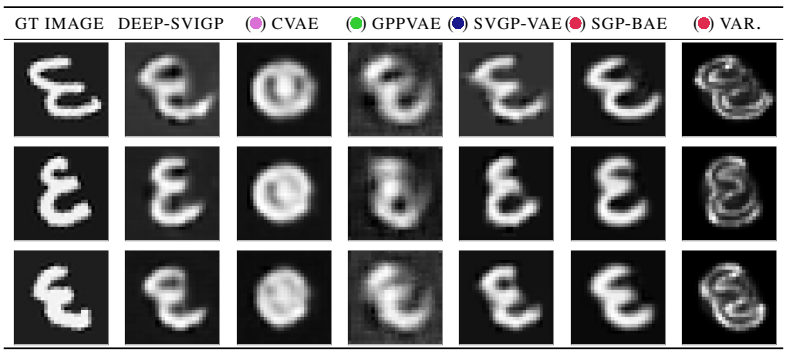
Figure 4: The posterior of the inducing inputs.

using a sufficient number of inducing points and operating in a Bayesian way, SGP-BAE obtains a distribution over lengthscales that is compatible with observed data. When using too few inducing points, the model tends to estimate a larger lengthscale. This is expected as the effective lengthscale of the observed process in the subspace spanned by these few inducing points is larger. We also observe that the more inducing points are used, the more confident the posterior over the lengthscale is. Our method also produces a sensible posterior distribution on the inducing inputs, as shown in Fig. 4. The estimated inducing inputs are evenly spaced over the time dimension, which is reasonable since the latent trajectories are generated from stationary GPs.

5.2. Conditional generation of rotated MNIST

In the next experiment, we consider a large-scale benchmark of conditional generation. We follow the experimental setup from (Casale et al., 2018; Jazbec et al., 2021), where they use a rotated MNIST dataset ($N = 4050$). In particular, we are given a set of images of digit three that have been rotated at different angles in $[0, 2\pi)$. Our goal is to generate a new image rotated at an unseen angle. As it is not trivial to apply full GP for such a large dataset, we omit GP-VAE (Pearce, 2019) and GP-BAE baselines. We consider the GPPVAE

Table 3: Conditionally generated MNIST images. The right most column depicts the epistemic uncertainty obtained by our SGP-BAE model.



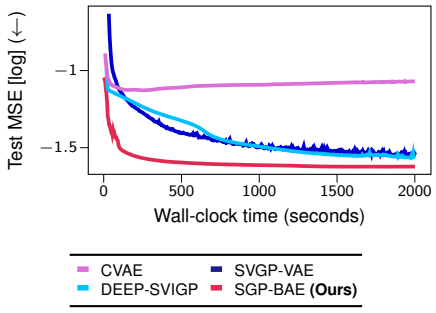


Figure 5: Comparison of test mean squared error (MSE) on the rotated MNIST dataset as function of training time.

model (Casale et al., 2018), which employs a low-rank approximation for the GP, and the SVGP-VAE model (Jazbec et al., 2021) as baselines. For both SVGP-VAE and SGP-BAE, we use a number of inducing points of $M = 32$ and a mini-batch size of $B = 256$. We also compare our method with the Conditional Variational Autoencoder (CVAE) of Sohn et al. (2015), which allows conditional generation tasks. Following Jazbec et al. (2021), we consider an extension of the sparse GP model (Hensman et al., 2013), named DEEP-SVIGP, that utilizes a deep likelihood parameterized by a neural network. As shown in Table 3, our SGP-BAE model generates images that are more visually appealing and most faithful to the ground truth compared to other approaches.

Performance on conditional generation. Table 4 presents the quantitative evaluation of the conditionally generated images in terms of MSE. Our SGP-BAE model clearly outperforms the other competing methods. It is worth noting that DEEP-SVIGP (Hensman et al., 2013) does not use an amortization mechanism, and its performance is considered to be an upper bound for that of SVGP-VAE. As discussed by Jazbec et al. (2021), DEEP-SVIGP can be used for conditional generation tasks, where the goal is to impose a single GP over the entire dataset, and therefore amortization is not necessary. However, this model cannot be used in tasks where inference has to be amortized across multiple GPs, such as learning interpretable data representations.

Computational efficiency. Similarly to the competing methods that use sparse approximations such as SVGP-VAE and DEEP-SVIGP, each iteration of GP-BAE involves the computation of the inverse covariance matrix, resulting in a time complexity of $\mathcal{O}(M^3)$. Fig. 5 shows the convergence in terms of test MSE for the competing methods and our SGP-BAE, trained for a fixed training time budget. We omit GPPVAE (Casale et al., 2018) from this comparison, as reported by Jazbec et al. (2021), which is significantly slower than the sparse methods. This result demonstrates that SGP-BAE converges remarkably faster in terms of wall-clock time while achieving better final predictive performance.

Table 4: Results on the rotated MNIST digit 3 dataset. Here, we report the mean and standard deviation computed from 4 runs.

	MODEL	MSE (↓)
	CVAE (Sohn et al., 2015)	0.0819 ± 0.0027
	GPPVAE (Casale et al., 2018)	0.0351 ± 0.0005
	SVGP-VAE (Jazbec et al., 2021)	0.0257 ± 0.0004
	SGP-BAE **(OURS)**	$\mathbf{0.0228} \pm \mathbf{0.0004}$
	DEEP-SVIGP (Jazbec et al., 2021)	0.0236 ± 0.0010

Epistemic uncertainty quantification. Unlike the competing methods, our SGP-BAE model can capture the epistemic uncertainty of the decoder thanks to the Bayesian treatment and the use of powerful inference methods. This can improve the quality of uncertainty quantification on reconstructed or generated images. As shown in the rightmost column of Table 3, SGP-BAE can provide sensible epistemic uncertainty quantification. Our model exhibits increased uncertainty for semantically and visually challenging pixels, such as the boundaries of the digits.

5.3. Missing data imputation

Next, we consider the task of imputing missing data on multi-output spatio-temporal datasets. We compare our method against Sparse GP-VAE (SGP-VAE) (Ashman et al., 2020) and multi-output GP methods such as Independent GPs (IGP) and Gaussian Process Autoregressive Regression (GPAR) (Requeima et al., 2019). Additionally, we consider the DSGP-BAE model, which is an extension of our SGP-BAE model endowed with 3-layer deep GP prior. For a fair comparison, we treat partially missing data as zeros to feed into the inference network (encoder) for SGP-VAE and our SGP-BAE model. We leave the adoption of partial inference networks (Ashman et al., 2020) to our model for future work. We follow the experimental setup of Requeima et al. (2019) and Ashman et al. (2020) and use two standard datasets for this comparison.

Table 5: A comparison between methods of multi-output GP models and GP autoencoders on the EEG and JURA datasets.

DATASET	METRIC	IGP	GPAR	SGP-VAE	SGP-BAE (OURS)	DSGP-BAE (OURS)
EEG	SMSE (↓)	1.70 ± 0.14	0.28 ± 0.00	0.52 ± 0.05	$\mathbf{0.22} \pm \mathbf{0.01}$	$\mathbf{0.21} \pm \mathbf{0.01}$
	NLL (↓)	2.59 ± 0.02	$\mathbf{1.68} \pm \mathbf{0.01}$	1.98 ± 0.02	1.96 ± 0.08	2.25 ± 0.13
JURA	MAE (↓)	0.57 ± 0.00	$\mathbf{0.42} \pm \mathbf{0.01}$	0.54 ± 0.01	$\mathbf{0.45} \pm \mathbf{0.03}$	$\mathbf{0.44} \pm \mathbf{0.02}$
	NLL (↓)	1563.42 ± 166.55	17.81 ± 1.06	1.02 ± 0.01	$\mathbf{0.91} \pm \mathbf{0.04}$	$\mathbf{0.85} \pm \mathbf{0.04}$

Electroencephalogram (EEG). This dataset comprises 256 measurements taken over one second. Each measurement consists of seven electrodes, FZ and F1-F6, placed on the patients scalp ($\mathbf{x}_n \in \mathbb{R}^1$, $\mathbf{y}_n \in \mathbb{R}^7$). The goal is to predict the last 100 samples for electrodes FZ, F1, and F2, given that the first 156 samples of FZ, F1, and F2 and the whole signals of F3-F6 are observed. Performance is measured with the standardized mean squared error (SMSE) and negative log-likelihood (NLL).

Jura. This is a geospatial dataset consisting of 359 measurements of the topsoil concentrations of three metals — Nickel, Zinc, and Cadmium — collected from a region of Swiss Jura ($\mathbf{x}_n \in \mathbb{R}^2$, $\mathbf{y}_n \in \mathbb{R}^3$). The dataset is split into a training set comprised of Nickel and Zinc measurements for all 359 locations and Cadmium measurements for just 259 locations. The task is to predict the Cadmium measurements at the remaining 100 locations conditioned on the observed training data. Performance is evaluated with the mean absolute error (MAE) and negative log-likelihood.

Table 5 compares the performance of SGP-BAE to the competing methods. As mentioned in Ashman et al. (2020), GPAR is the state-of-the-art method for these datasets. We find that our SGP-BAE and DSGP-BAE models perform comparably with GPAR on the EEG dataset but better on the JURA dataset. A significant advantage of GP autoencoder methods is that they model P outputs using only C latent GPs, while GPAR uses P GPs. This can be beneficial when the dimensionality of the data, P, is very high. Similarly to the previous experiments, SGP-BAE consistently performs better than variational approximation-based methods such as SGP-VAE (Ashman et al., 2020). As expected, IGP is the worst-performing method due to its inability to model the correlations between output variables.

We find that the utilization of deep Gaussian process (DGP) priors yields only marginal improvement on the EEG and JURA datasets, as shown in Table 5. In addition, we observe that DSGP-BAE just performs comparably to SGP-BAE on the moving ball dataset. This can be attributed to the fact that the correlation between the latent variables of these datasets is sufficiently simple to be modeled using shallow GPs. For instance, in the experiments conducted on the moving ball dataset, the data is generated from a GP, following the setup used in the previous work (Pearce, 2019; Jazbec et al., 2021). Nevertheless, when dealing with more complex datasets, we believe that the flexibility offered by DGPs can prove beneficial for modeling intricate patterns (e.g., discontinuities or strong non-stationarities) of the latent process.

6. Conclusions

We have introduced our novel SGP-BAE that integrates fully Bayesian sparse GPs on the latent space of Bayesian autoencoders. Our proposed model is generic, as it allows an arbitrary GP kernel and deep GPs. The inference for this model is carried out by a powerful and scalable SGHMC sampler. Through a rigorous experimental campaign, we have demonstrated the excellent performance of SGP-BAE on a wide range of representation learning and generative modeling tasks.

Limitations and future works. While our model's ability to learn disentangled representations has been demonstrated through empirical evidence, there is a need to establish a theoretical grounding for the disentanglement of its latent space. Furthermore, it would be useful to study the amortization gap (Cremer et al., 2018; Krishnan et al., 2018; Marino et al., 2018) of our model. Additionally, exploring more informative priors for the decoder's parameters (Tran et al., 2021), beyond the isotropic Gaussian prior used in this work, is also worthwhile. There are potential avenues for future research. One of them is the fully Bayesian treatment of the auxiliary data $\mathbf{X}$. This approach can be beneficial when the auxiliary data is unavailable or contains missing values. In addition, it would be interesting to apply the model to downstream applications where modeling correlations between data points and uncertainty quantification are required, such as in bioinformatics and climate modeling.

Acknowledgements

We thank Simone Rossi and Dimitrios Milios for helpful discussions. BS acknowledges support from the National Institutes of Health (NIH) under grant NIH-R01MH115697. SM acknowledges support from the National Science Foundation (NSF) under an NSF CAREER Award, award numbers 2003237 and 2007719, by the Department of Energy under grant DE-SC0022331, by the HPI Research Center in Machine Learning and Data Science at UC Irvine, and by gifts from Qualcomm and Disney. MF gratefully acknowledges support from the AXA Research Fund and the Agence Nationale de la Recherche (grant ANR-18-CE46-0002 and ANR-19-P3IA-0002).

References

Ashman, M., So, J., Tebbutt, W., Fortuin, V., Pearce, M., and Turner, R. E. Sparse Gaussian Process Variational Autoencoders. *arXiv preprint arXiv:2010.10177*, 2020.

Bauer, M. and Mnih, A. Resampled Priors for Variational Autoencoders. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pp. 66–75. PMLR, 2019.

Bengio, Y., Courville, A., and Vincent, P. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35 (8):1798–1828, 2013.

Bishop, C. M. *Pattern Recognition and Machine learning*, volume 4. Springer, 2006.

Bonilla, E. V., Krauth, K., and Dezfouli, A. Generic Inference in Latent Gaussian Process Models. *Journal of Machine Learning Research*, 20(117):1–63, 2019.

Casale, F. P., Dalca, A., Saglietti, L., Listgarten, J., and Fusi, N. Gaussian Process Prior Variational Autoencoders. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Chen, T., Fox, E., and Guestrin, C. Stochastic Gradient Hamiltonian Monte Carlo. In *Proceedings of the 31st International Conference on Machine Learning, ICML 2014*, Proceedings of Machine Learning Research, pp. 1683–1691, Bejing, China, 22–24 Jun 2014. PMLR.

Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. Variational Lossy Autoencoder. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Cottrell, G. W., Munro, P., and Zipser, D. Image Compression by Back Propagation: A Demonstration of Extensional Programming. *Models of Cognition*, pp. 208–240, 1989.

Cremer, C., Li, X., and Duvenaud, D. Inference Suboptimality in Variational Autoencoders. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Damianou, A. and Lawrence, N. D. Deep Gaussian Processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pp. 207–215, Scottsdale, Arizona, USA, 29 Apr–01 May 2013. PMLR.

Daxberger, E. A. and Hernández-Lobato, J. M. Bayesian Variational Autoencoders for Unsupervised Out-of-Distribution Detection. *arXiv preprint arXiv:1912.05651*, 2019.

Dilokthanakul, N., Mediano, P. A., Garnelo, M., Lee, M. C., Salimbeni, H., Arulkumaran, K., and Shanahan, M. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*, 2016.

Feng, Y., Wang, D., and Liu, Q. Learning to Draw Samples with Amortized Stein Variational Gradient Descent. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press, 2017.

Fortuin, V., Baranchuk, D., Raetsch, G., and Mandt, S. GP-VAE: Deep Probabilistic Time Series Imputation. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 1651–1661. PMLR, 26–28 Aug 2020.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S., and Teh, Y. W. Neural Processes. *arXiv preprint arXiv:1807.01622*, 2018.

Gelman, A. and Rubin, D. B. Inference from Iterative Simulation using Multiple Sequences. *Statistical Science*, 7(4):457–472, 1992.

Glazunov, M. and Zarras, A. Do bayesian variational autoencoders know what they dont know? In *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pp. 718–727. PMLR, 01–05 Aug 2022.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian Processes for Big Data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, pp. 282290, Arlington, Virginia, USA, 2013. AUAI Press.

Hensman, J., de G. Matthews, A. G., and Ghahramani, Z. Scalable Variational Gaussian Process Classification. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS*

2015, San Diego, California, USA, May 9-12, 2015, volume 38 of *JMLR Workshop and Conference Proceedings*. JMLR.org, 2015a.

Hensman, J., Matthews, A. G., Filippone, M., and Ghahramani, Z. MCMC for Variationally Sparse Gaussian Processes. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015b.

Ho, J., Jain, A., and Abbeel, P. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020.

Izmailov, P., Vikram, S., Hoffman, M. D., and Wilson, A. G. G. What Are Bayesian Neural Network Posteriors Really Like? In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4629–4640. PMLR, 18–24 Jul 2021.

Jazbec, M., Ashman, M., Fortuin, V., Pearce, M., Mandt, S., and Rätsch, G. Scalable Gaussian Process Variational Autoencoders . In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 3511–3519. PMLR, 13–15 Apr 2021.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An Introduction to Variational Methods for Graphical Models. *Machine learning*, 37(2):183–233, 1999.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014.

Korattikara Balan, A., Rathod, V., Murphy, K. P., and Welling, M. Bayesian Dark Knowledge. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Krishnan, R., Liang, D., and Hoffman, M. On the Challenges of Learning with Inference Networks on Sparse, High-dimensional Data. In Storkey, A. and Perez-Cruz, F. (eds.), *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 143–151. PMLR, 09–11 Apr 2018.

Lalchand, V., Bruinsma, W., Burt, D., and Rasmussen, C. E. Sparse Gaussian Process Hyperparameters: Optimize or Integrate? In *Advances in Neural Information Processing Systems*, volume 35, pp. 16612–16623. Curran Associates, Inc., 2022a.

Lalchand, V., Ravuri, A., and Lawrence, N. D. Generalised GPLVM with Stochastic Variational Inference . In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 7841–7864. PMLR, 28–30 Mar 2022b.

Lawrence, N. D. Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models. *Journal of Machine Learning Research*, 6:1783–1816, 2005.

Li, Y., Turner, R. E., and Liu, Q. Approximate Inference with Amortised MCMC. *arXiv preprint arXiv:1702.08343*, 2017.

MacKay, D. J. Bayesian Methods for Backpropagation Networks. In *Models of neural networks III*, pp. 211–254. Springer, 1996.

Mackay, D. J. C. *Bayesian Methods for Adaptive Models*. PhD thesis, California Institute of Technology, USA, 1992. UMI Order No. GAX92-32200.

Marino, J., Yue, Y., and Mandt, S. Iterative Amortized Inference. In *International Conference on Machine Learning*, pp. 3403–3412. PMLR, 2018.

Miani, M., Warburg, F., Moreno-Muñoz, P., Skafte, N., and Hauberg, S. r. Laplacian Autoencoders for Learning Stochastic Representations. In *Advances in Neural Information Processing Systems*, volume 35, pp. 21059–21072. Curran Associates, Inc., 2022.

Nalisnick, E. T. and Smyth, P. Stick-Breaking Variational Autoencoders. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Neal, R. M. *Bayesian Learning for Neural Networks (Lecture Notes in Statistics)*. Springer, 1 edition, August 1996.

Neal, R. M. *MCMC Using Hamiltonian Dynamics*, chapter 5. CRC Press, 2011. doi: 10.1201/b10905-7.

Pearce, M. The Gaussian Process Prior VAE for Interpretable Latent Dynamics from Pixels. In *Symposium on Advances in Approximate Bayesian Inference, AABI 2019, Vancouver, BC, Canada, December 8, 2019*, volume 118 of *Proceedings of Machine Learning Research*, pp. 1–12. PMLR, 2019.

Quiñonero-Candela, J. and Rasmussen, C. E. A Unifying View of Sparse Approximate Gaussian Process Regression. *Journal of Machine Learning Research*, 6(65):1939–1959, 2005.

Ramchandran, S., Tikhonov, G., Kujanpää, K., Koskinen, M., and Lähdesmäki, H. Longitudinal Variational Autoencoder . In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 3898–3906. PMLR, 13–15 Apr 2021.

Rasmussen, C. E. and Williams, C. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Requeima, J., Tebbutt, W., Bruinsma, W., and Turner, R. E. The Gaussian Process Autoregressive Regression Model (GPAR). In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 1860–1869. PMLR, 16–18 Apr 2019.

Rezende, D. and Mohamed, S. Variational Inference with Normalizing Flows. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1530–1538, Lille, France, 07–09 Jul 2015. PMLR.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014*, volume 32 of *Proceeding of Machine Learning Research*, pp. 1278–1286, Beijing, China, 21-26 June 2014. PMLR.

Rossi, S., Heinonen, M., Bonilla, E., Shen, Z., and Filippone, M. Sparse Gaussian Processes Revisited: Bayesian Approaches to Inducing-Variable Approximations . In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 1837–1845. PMLR, 13–15 Apr 2021.

Salimbeni, H. and Deisenroth, M. Doubly Stochastic Variational Inference for Deep Gaussian Processes. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Shi, J., Khan, M. E., and Zhu, J. Scalable Training of Inference Networks for Gaussian-Process Models. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5758–5768. PMLR, 09–15 Jun 2019.

Snelson, E. and Ghahramani, Z. Sparse Gaussian Processes using Pseudo-inputs. In *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005.

Sohn, K., Lee, H., and Yan, X. Learning Structured Output Representation using Deep Conditional Generative Models. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Springenberg, J. T., Klein, A., Falkner, S., and Hutter, F. Bayesian Optimization with Robust Bayesian Neural Networks. In *Advances in Neural Information Processing Systems*, volume 29, pp. 4134–4142. Curran Associates, Inc., 2016.

Titsias, M. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pp. 567–574, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.

Tomczak, J. M. Deep Generative Modeling. In *Deep Generative Modeling*, pp. 1–12. Springer, 2022.

Tomczak, J. M. and Welling, M. VAE with a VampPrior. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, volume 84 of *Proceedings of Machine Learning Research*, pp. 1214–1223. PMLR, 2018.

Tran, B.-H., Rossi, S., Milios, D., Michiardi, P., Bonilla, E. V., and Filippone, M. Model Selection for Bayesian Autoencoders. In *Advances in Neural Information Processing Systems*, volume 34, pp. 19730–19742. Curran Associates, Inc., 2021.

Tran, B.-H., Rossi, S., Milios, D., and Filippone, M. All You Need is a Good Functional Prior for Bayesian Deep Learning. *Journal of Machine Learning Research*, 23 (74):1–56, 2022.

van der Maaten, L. and Hinton, G. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86): 2579–2605, 2008.

Wang, D. and Liu, Q. Learning to Draw Samples: With Application to Amortized MLE for Generative Adversarial Learning. *arXiv preprint arXiv:1611.01722*, 2016.

Wang, K.-C., Vicol, P., Lucas, J., Gu, L., Grosse, R., and Zemel, R. Adversarial Distillation of Bayesian Neural Network Posteriors. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5190–5199. PMLR, 10–15 Jul 2018.

Wilson, A. G. and Izmailov, P. Bayesian Deep Learning and a Probabilistic Perspective of Generalization. In *Advances in Neural Information Processing Systems*, volume 33, pp. 4697–4708. Curran Associates, Inc., 2020.

Yang, Y., Mandt, S., and Theis, L. An Introduction to Neural Data Compression. *arXiv preprint arXiv:2202.06533*, 2022.

Zhang, C., Bütepage, J., Kjellström, H., and Mandt, S. Advances in Variational Inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.

Zhang, R., Li, C., Zhang, J., Chen, C., and Wilson, A. G. Cyclical Stochastic Gradient MCMC for Bayesian Deep Learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Zhu, H., Rodas, C. B., and Li, Y. Markovian Gaussian Process Variational Autoencoders. *arXiv preprint arXiv:2207.05543*, 2022.

A. A taxonomy of latent variable models

By considering the characteristics of the prior distribution on latent variables, the likelihood function, input dependencies, and Bayesian treatments, we can draw connections between our proposed models and other latent variable models. Fig. 6 summarizes these relationships. Here, we assume an isotropic Gaussian likelihood with precision β for the high-dimensional observed data $\mathbf{y}_n$ as used in our experiments.

Probabilistic principal component analysis (PCA) (Bishop, 2006) is a simple latent variable model that imposes an isotropic Gaussian prior over the latent space and linear mapping from the latent variables to the observed data. Variational Autoencoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014) build upon this by introducing a nonlinear parametric mapping to the observed data, while Gaussian process latent variable models (GPLVMs) utilize a nonparametric Gaussian Process (GP) mapping. Recently, Lalchand et al. (2022b) extended GPLVMs in a fully Bayesian manner using stochastic variational inference (VI). Conditional Variational Autoencoder (CVAE) (Sohn et al., 2015) is an extension of VAEs that utilizes an input-dependent prior in the latent space for conditional generation tasks. However, this model does not account for correlations between latent representations. Gaussian Process VAEs (Casale et al., 2018; Pearce, 2019; Jazbec et al., 2021) overcome this problem by imposing GP priors on the latent space. Our model, Sparse Gaussian Process Bayesian Autoencoder (SGP-BAE), further enhances the modeling capabilities of these models by using a fully Bayesian approach for the latent variables, decoder, and GP hyperparameters in a fully Bayesian manner, and decoupling the model from the variational inference formulation.

Latent neural processes (Garnelo et al., 2018) can be seen as an extension of CVAE. However, this model follows a meta-learning approach and splits the data into a context set, $\left\{\mathbf{x}_n^{[C]}, \mathbf{y}_n^{[C]}\right\}_{n=1}^{N_C}$, and a target set, $\left\{\mathbf{x}_n^{[T]}, \mathbf{y}_n^{[T]}\right\}_{n=1}^{N_T}$. This model uses a global latent variable $\mathbf{z}$ to capture the global properties of the context data. The likelihood is conditioned on both new target input $\mathbf{x}_n^{[T]}$ and the global latent variable $\mathbf{z}$.

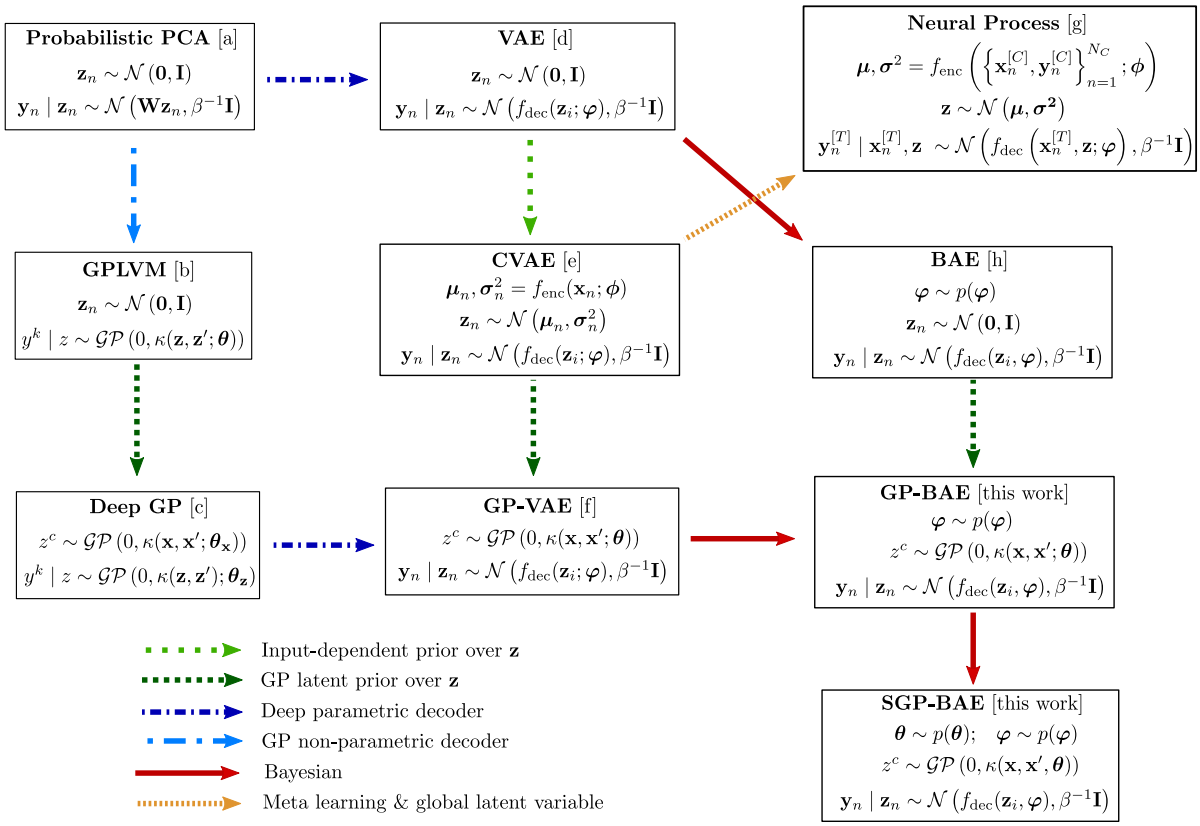


Figure 6: Connections between our proposed models and other latent variables models. References are [a] for Bishop (2006), [b] for Lawrence (2005), [c] for Damianou & Lawrence (2013), [d] for Kingma & Welling (2014); Rezende et al. (2014), [e] for Sohn et al. (2015), [f] for Casale et al. (2018); Pearce (2019); Jazbec et al. (2021), [g] for Garnelo et al. (2018), and [h] for this work and Kingma & Welling (2014); Daxberger & Hernández-Lobato (2019).

B. Details on stochastic gradient Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) (Neal, 2011) is a highly-efficient Markov Chain Monte Carlo (MCMC) method used to generate samples $\boldsymbol{\theta}$ from a potential energy function $U(\boldsymbol{\theta})$. HMC introduces auxiliary momentum variables $\mathbf{r}$ and samples are then generated from the joint distribution $p(\boldsymbol{\theta}, \mathbf{r})$ using the Hamiltonian dynamics:

$$\begin{cases} d\boldsymbol{\theta} &= \mathbf{M}^{-1}\mathbf{r}dt, \\ d\mathbf{r} &= -\nabla U(\boldsymbol{\theta})dt, \end{cases} \tag{12}$$

where $\mathbf{M}$ is an arbitrary mass matrix that serves as a preconditioner. The continuous system described by HMC is approximated using η-discretized numerical integration, and subsequent Metropolis steps are applied to account for errors that may result from this approximation.

However, HMC becomes computationally inefficient when applied to large datasets, as it requires the calculation of the gradient $\nabla U(\boldsymbol{\theta})$ on the entire dataset. To address this issue, (Chen et al., 2014) proposed an extension of HMC called stochastic gradient Hamiltonian Monte Carlo (SGHMC), which uses a noisy but unbiased estimate of the gradient $\nabla \tilde{U}(\boldsymbol{\theta})$ computed from a mini-batch of the data. The discretized Hamiltonian dynamics are then updated using this estimate of the gradient as follows:

$$\begin{cases} \Delta\boldsymbol{\theta} &= \mathbf{M}^{-1}\mathbf{r}, \\ \Delta\mathbf{r} &= -\eta\nabla\tilde{U}(\boldsymbol{\theta}) - \eta\mathbf{C}\mathbf{M}^{-1}\mathbf{r} + \mathcal{N}(0, 2\eta(\mathbf{C} - \tilde{\mathbf{B}})), \end{cases} \tag{13}$$

where η is the step size, $\mathbf{C}$ is a user-defined friction matrix, $\tilde{\mathbf{B}}$ is the estimate for the noise of the gradient evaluation. To select suitable values for these hyperparameters, we use a scale-adapted version of SGHMC proposed by Springenberg et al. (2016), in which the hyperparameters are automatically adjusted during a burn-in phase. After this period, the values of the hyperparameters are fixed.

Estimating $\mathbf{M}$. In our implementation of SGHMC, we set the mass matrix $\mathbf{M}^{-1} = \operatorname{diag}\left(\hat{V}_{\boldsymbol{\theta}}^{-1/2}\right)$, where $\hat{V}_{\boldsymbol{\theta}}$ is an estimate of the uncentered variance of the gradient, $\hat{V}_{\boldsymbol{\theta}} \approx \mathbb{E}[(\nabla\tilde{U}(\boldsymbol{\theta}))^2]$. which can be estimated by using the exponential moving average as follows:

$$\Delta\hat{V}_{\boldsymbol{\theta}} = -\tau^{-1}\hat{V}_{\boldsymbol{\theta}} + \tau^{-1}\nabla(\tilde{U}(\boldsymbol{\theta}))^2, \tag{14}$$

where τ is a parameter vector that specifies the moving average windows. This parameter can be chosen adaptively using the method proposed by Springenberg et al. (2016) as follows:

$$\Delta\tau = -g_{\boldsymbol{\theta}}^2\hat{V}_{\boldsymbol{\theta}}^{-1}\tau + 1, \quad \text{and,} \quad \Delta g_{\boldsymbol{\theta}} = -\tau^{-1}g_{\boldsymbol{\theta}} + \tau^{-1}\nabla\tilde{U}(\boldsymbol{\theta}), \tag{15}$$

where $g_{\boldsymbol{\theta}}$ is a smoothed estimate of the gradient $\nabla U(\boldsymbol{\theta})$.

Estimating $\tilde{\mathbf{B}}$. Ideally, the estimate of the noise of the gradient evaluation, $\tilde{\mathbf{B}}$, should be the empirical Fisher information matrix of $U(\boldsymbol{\theta})$. However, this matrix is computationally expensive to calculate, so we instead use a diagonal approximation given by $\tilde{\mathbf{B}} = \frac{1}{2}\eta\hat{V}_{\boldsymbol{\theta}}$. This approximation is already obtained from the step of estimating $\mathbf{M}$.

Choosing $\mathbf{C}$. In practice, it is common to set the friction matrix $\mathbf{C}$ to be equal to the identity matrix, i.e. $\mathbf{C} = C\mathbf{I}$, which means that the same independent noise is applied to each element of $\boldsymbol{\theta}$.

The discretized Hamiltonian dynamics. By substituting $\mathbf{v} := \eta\hat{V}_{\boldsymbol{\theta}}^{-1/2}\mathbf{r}$, the dynamics in Eq. 13 becomes

$$\begin{cases} \Delta\boldsymbol{\theta} &= \mathbf{v}, \\ \Delta\mathbf{v} &= -\eta^2\hat{V}_{\boldsymbol{\theta}}^{-1/2}\nabla\tilde{U}(\boldsymbol{\theta}) - \eta C\hat{V}_{\boldsymbol{\theta}}^{-1/2}\mathbf{v} + \mathcal{N}(0, 2\eta^3 C\hat{V}_{\boldsymbol{\theta}}^{-1} - \eta^4\mathbf{I}). \end{cases} \tag{16}$$

Following (Springenberg et al., 2016), we choose C such that $\eta C\hat{V}_{\boldsymbol{\theta}}^{-1/2} = \alpha\mathbf{I}$. This is equivalent to using a constant momentum coefficient of α. The final discretized dynamics are then

$$\begin{cases} \Delta\boldsymbol{\theta} &= \mathbf{v}, \\ \Delta\mathbf{v} &= -\eta^2\hat{V}_{\boldsymbol{\theta}}^{-1/2}\nabla\tilde{U}(\boldsymbol{\theta}) - \alpha\mathbf{v} + \mathcal{N}(0, 2\eta^2\alpha\hat{V}_{\boldsymbol{\theta}}^{-1/2} - \eta^4\mathbf{I}). \end{cases} \tag{17}$$

C. Details of the scalable sampling objective for sparse Gaussian processes

GPs (Rasmussen & Williams, 2006) are one of the main workhorses of Bayesian nonparametric statistics and machine learning, as they provide a flexible and powerful tool for imposing a prior distribution over functions:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta})), \tag{18}$$

where $f(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}$ maps D-dimensional inputs into one-dimensional outputs. GPs are fully characterized by their mean and their covariance:

$$\mathbb{E}[f(\mathbf{x})] = m(\mathbf{x}), \qquad \text{cov}[f(\mathbf{x}), f(\mathbf{x}')] = \kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}), \tag{19}$$

where $m : \mathbb{R}^D \to \mathbb{R}$ is the mean and $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is the kernel function with hyperparameters $\boldsymbol{\theta}$. GPs indeed can be viewed as an extension of a multivariate Gaussian distribution to infinitely many dimensions. For any fixed set of inputs $\mathbf{X} \in \mathbb{R}^{N \times D}$, the realizations of functions with a GP prior at these inputs, denoted by $\mathbf{f} \in \mathbb{R}^N$, follow the Gaussian distribution:

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{xx} \,|\, \boldsymbol{\theta}}). \tag{20}$$

Here, we assume a zero-mean GP prior and $\mathbf{K}_{\mathbf{XX} \,|\, \boldsymbol{\theta}}$ is the covariance matrix obtained by evaluating $\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta})$ over all input pairs $\mathbf{x}_i$, $\mathbf{x}_j$ (we will drop the explicit parameterization on $\boldsymbol{\theta}$ for the sake of notation). From now on, in order to keep the notation uncluttered, we focus on a single channel of latent space of autoencoders (AEs). We assume that the latent codes $\mathbf{Z}$ are stochastic realizations based on $\mathbf{f}$ and additive Gaussian noise i.e. $\mathbf{Z} \,|\, \mathbf{f} \sim \mathcal{N}(\mathbf{f}, \sigma^2 \mathbf{I})$. We further assume a full factorization $p(\mathbf{Z} \,|\, \mathbf{f}; \sigma^2 \mathbf{I}) = \prod_{n=1}^{N} p(\mathbf{z}_n \,|\, f_n; \sigma^2)$.

Though GPs provide an elegant mechanism to handle uncertainties, their computational complexity grows cubically with the number of inputs, i.e. $\mathcal{O}(N^3)$. This problem is commonly tackled by sparse GPs, which are a family of approximate models that address the scalability issue by introducing a set of M inducing variables $\mathbf{u} = (u_1, ..., u_M)^\top \in \mathbb{R}^{M \times 1}$ at the corresponding inducing inputs $\mathbf{S} = [\mathbf{s}_1^\top, ..., \mathbf{s}_M^\top]^\top \in \mathbb{R}^{M \times D}$ with $u_m \equiv f(\mathbf{s}_m)$. The inducing points $\mathbf{S}$ can be interpreted as a compressed version of the training data where the number of inducing points M acts as a trade-off parameter between the goodness of the approximation and scalability. The inducing variables are assumed to be drawn from the same GP as the original process, i.e.:

$$p(\mathbf{f}, \mathbf{u}) = p(\mathbf{u}) p(\mathbf{f} \,|\, \mathbf{u}), \text{with} \tag{21}$$

$$p(\mathbf{u}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{ss}}), \tag{22}$$

$$p(\mathbf{f} \,|\, \mathbf{u}) = \mathcal{N}(\mathbf{K}_{\mathbf{xs}} \mathbf{K}_{\mathbf{ss}}^{-1} \mathbf{u}, \mathbf{K}_{\mathbf{xx}} - \mathbf{K}_{\mathbf{xs}} \mathbf{K}_{\mathbf{ss}}^{-1} \mathbf{K}_{\mathbf{sx}}), \tag{23}$$

where the covariance matrices $\mathbf{K}_{\mathbf{ss}}, \mathbf{K}_{\mathbf{xs}}$ are computed between the elements in $\mathbf{S}$ and $\{\mathbf{X}, \mathbf{S}\}$, respectively, and $\mathbf{K}_{\mathbf{sx}} = \mathbf{K}_{\mathbf{sx}}^\top$.

There is a line of works using variational techniques for sparse GPs priors for VAEs (Jazbec et al., 2021; Ashman et al., 2020). More specifically, they rely on the variational framework proposed by Titsias (2009); Hensman et al. (2013), enabling the use of GP priors on very large datasets. However, the posterior of the inducing variables $\mathbf{u}$ under this framework involves constraining to have a parametric form (usually a Gaussian).

In this work, we take a different route as we treat the inducing variables $\mathbf{u}$, inducing inputs $\mathbf{S}$ as well as the kernel hyperparameters $\boldsymbol{\theta}$ in a fully Bayesian way. As discussed in the main paper, we wish to infer these variables together with the decoder parameters and the latent codes by using a powerful and scalable SGHMC (Chen et al., 2014) sampler. To do so, the sampling objective Eq. 9 should be decomposed over all data points. The main obstacle is the joint distribution $p(\mathbf{Z}, \mathbf{u} \,|\, \boldsymbol{\theta}) = \mathbb{E}_{p(\mathbf{f} \,|\, \mathbf{u})}[p(\mathbf{Z} \,|\, \mathbf{f}; \sigma^2 \mathbf{I})]$, which has a complicated form due to the expectation under the conditional $p(\mathbf{f} \,|\, \mathbf{u})$. As discussed by Rossi et al. (2021), this problem can be solved effectively by the fully independent training conditionals (FITC; see Quiñonero-Candela & Rasmussen, 2005), i.e., parameterizing $p(\mathbf{f} \,|\, \mathbf{u})$ as follows:

$$p(\mathbf{f} \,|\, \mathbf{u}) \approx \mathcal{N}(\mathbf{f}; \mathbf{K}_{\mathbf{xs}} \mathbf{K}_{\mathbf{ss}}^{-1} \mathbf{u}, \text{diag}[\mathbf{K}_{\mathbf{xx}} - \mathbf{K}_{\mathbf{xs}} \mathbf{K}_{\mathbf{ss}}^{-1} \mathbf{K}_{\mathbf{sx}}]) \tag{24}$$

$$= \prod_{n=1}^{N} p(f_n \,|\, \mathbf{u}) = \prod_{n=1}^{N} \mathcal{N}(f_n; \tilde{\mu}_n, \tilde{\sigma}_n^2), \tag{25}$$

where

$$\tilde{\mu}_n = \kappa(\mathbf{x}_n, \mathbf{S})\mathbf{K}_{\mathbf{ss}}^{-1}\mathbf{u}, \tag{26}$$

$$\tilde{\sigma}_n^2 = \kappa(\mathbf{x}_n, \mathbf{x}_n) - \kappa(\mathbf{x}_n, \mathbf{S})\mathbf{K}_{\mathbf{ss}}^{-1}\kappa(\mathbf{S}, \mathbf{x}_n). \tag{27}$$

We then can decompose the log-joint distribution over all data points as follows:

$$\begin{aligned}
\log p(\mathbf{Z}, \mathbf{u} \mid \boldsymbol{\theta}) &= \log \mathbb{E}_{p(\mathbf{f} \mid \mathbf{u}, \boldsymbol{\theta})}[p(\mathbf{z} \mid \mathbf{f}; \sigma^2\mathbf{I})] && (28)\\
&= \log \int p(\mathbf{f} \mid \mathbf{u}, \boldsymbol{\theta}) p(\mathbf{Z} \mid \mathbf{f}, \sigma^2\mathbf{I}) d\mathbf{f} && (29)\\
&= \log \int \prod_{n=1}^{N} p(\mathbf{z}_n \mid f_n; \sigma^2) p(f_n \mid \mathbf{u}, \boldsymbol{\theta}) df_1 ... df_n && (30)\\
&= \log \prod_{n=1}^{N} \int \mathcal{N}(\mathbf{z}_n; f_n, \sigma^2)\mathcal{N}(f_n; \tilde{\mu}_n, \tilde{\sigma}_n^2) df_n && (31)\\
&= \log \prod_{n=1}^{N} \mathcal{N}(\mathbf{z}_n; \tilde{\mu}_n, \tilde{\sigma}_n^2 + \sigma^2) && (32)\\
&= \sum_{n=1}^{N} \log \mathcal{N}(\mathbf{z}_n; \tilde{\mu}_n, \tilde{\sigma}_n^2 + \sigma^2), && (33)
\end{aligned}$$

where $\tilde{\mu}_n, \tilde{\sigma}_n^2$ are given by Eq. 26 and Eq. 27, respectively.

In this work, we adopt the approach proposed by Hensman et al. (2015b), which involves sampling the hyperparameters $\boldsymbol{\theta}$ and $\mathbf{u}$ jointly. To achieve sampling efficiency, a whitening representation is utilized, where the inducing variables are reparameterized as $\mathbf{u} = \mathbf{L}_{\mathbf{ss}}\boldsymbol{\nu}$, with $\mathbf{K}_{\mathbf{ss}} = \mathbf{L}_{\mathbf{ss}}\mathbf{L}_{\mathbf{ss}}^{\top}$. Subsequently, the sampling process involves obtaining samples from the joint posterior distribution over $\boldsymbol{\nu}$ and $\boldsymbol{\theta}$.

D. Details of the extension to deep Gaussian processes

We assume a deep Gaussian process prior $f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)}$ (Damianou & Lawrence, 2013), where each $f^{(l)}$ is a GP. For the sake of notation, we use $\boldsymbol{\Psi}^{(l)}$ to indicate the set of kernel hyperparameters and inducing inputs of the l-th layer and $\mathbf{f}^{(0)}$ as the input $\mathbf{X}$. We additionally denote $\boldsymbol{\Psi} = \{\boldsymbol{\varphi}\} \cup \left\{\mathbf{u}^{(l)}, \boldsymbol{\Psi}^{(l)}\right\}_{l=1}^{L}$, where $\boldsymbol{\varphi}$ is the decoder's parameters.

The full joint distribution is as follows:

$$p\left(\boldsymbol{\Psi}, \{\mathbf{f}^{(l)}\}_{l=1}^{L}, \mathbf{Z}, \mathbf{Y} \,\middle|\, \mathbf{X}\right) = p(\boldsymbol{\Psi}) \underbrace{\prod_{l=1}^{L} p\left(\mathbf{f}^{(l)} \mid \mathbf{f}^{(l-1)}, \boldsymbol{\Psi}\right) p\left(\mathbf{Z} \,\middle|\, \mathbf{f}^{(L)}; \sigma^2\mathbf{I}\right)}_{\text{Deep GP prior}} p(\mathbf{Y} \mid \mathbf{Z}, \boldsymbol{\varphi}). \tag{34}$$

Here we omit the dependency on $\mathbf{X}$ for notational brevity.

We wish to infer the posterior over $\boldsymbol{\Psi}$ and the latent variables $\mathbf{Z}$. To do this, the hidden layers $\mathbf{f}^{(l)}$ have to be marginalized and propagated up to the final layer L (Salimbeni & Deisenroth, 2017). In particular, the log posterior is as follows:

$$\log p(\boldsymbol{\Psi}, \mathbf{Z} \mid \mathbf{Y}, \mathbf{X}) = \log p(\boldsymbol{\Psi}) + \log \mathbb{E}_{p\left(\{\mathbf{f}^{(l)}\}_{l=1}^{L} \,\middle|\, \{\mathbf{u}^{(l)}, \boldsymbol{\Psi}^{(l)}\}_{l=1}^{L}\right)} p\left(\mathbf{Z} \,\middle|\, \mathbf{f}^{(L)}; \sigma^2\mathbf{I}\right) + \log p(\mathbf{Y} \mid \mathbf{Z}, \boldsymbol{\varphi}) - \log C, \tag{35}$$

where C is a normalizing constant.

The above posterior is intractable, but we have obtained the form of its (un-normalized) log-joint, from which we can sample using the HMC method. Unfortunately, the expectation term is intractable, but we can estimate it using Monte Carlo

sampling as follows:

$$\log \mathbb{E}_{p\left(\left\{\mathbf{f}^{(l)}\right\}_{l=1}^{L} \,\middle|\, \left\{\mathbf{u}^{(l)}, \boldsymbol{\Psi}^{(l)}\right\}_{l=1}^{L}\right)} p\left(\mathbf{Z} \,\middle|\, \mathbf{f}^{(L)}; \sigma^2 \mathbf{I}\right) \approx$$

$$\approx \log \mathbb{E}_{p\left(\left\{\mathbf{f}^{(l)}\right\}_{l=2}^{L} \,\middle|\, \widetilde{\mathbf{f}}^{(1)}, \left\{\mathbf{u}^{(l)}, \boldsymbol{\Psi}^{(l)}\right\}_{l=2}^{L}\right)} p\left(\mathbf{Z} \,\middle|\, \mathbf{f}^{(L)}; \sigma^2 \mathbf{I}\right), \qquad \widetilde{\mathbf{f}}^{(1)} \sim p\left(\mathbf{f}^{(1)} \,\middle|\, \mathbf{u}^{(1)}, \boldsymbol{\Psi}^{(1)}, \mathbf{f}^{(0)}\right), \tag{36}$$

$$\approx \log \mathbb{E}_{p\left(\left\{\mathbf{f}^{(l)}\right\}_{l=3}^{L} \,\middle|\, \widetilde{\mathbf{f}}^{(2)}, \left\{\mathbf{u}^{(l)}, \boldsymbol{\Psi}^{(l)}\right\}_{l=3}^{L}\right)} p\left(\mathbf{Z} \,\middle|\, \mathbf{f}^{(L)}; \sigma^2 \mathbf{I}\right), \quad \widetilde{\mathbf{f}}^{(2)} \sim p\left(\mathbf{f}^{(2)} \,\middle|\, \mathbf{u}^{(2)}, \boldsymbol{\Psi}^{(2)}, \widetilde{\mathbf{f}}^{(1)}\right), \tag{37}$$

$$\approx \dots$$

$$\approx \log \mathbb{E}_{p\left(\mathbf{f}^{(L)} \,\middle|\, \widetilde{\mathbf{f}}^{(L-1)}, \mathbf{u}^{(L)}, \boldsymbol{\Psi}^{(L)}\right)} p\left(\mathbf{Z} \,\middle|\, \mathbf{f}^{(L)}; \sigma^2 \mathbf{I}\right), \quad \widetilde{\mathbf{f}}^{(L-1)} \sim p\left(\mathbf{f}^{L-1} \,\middle|\, \mathbf{u}^{(L-1)}, \boldsymbol{\Psi}^{(L-1)}, \widetilde{\mathbf{f}}^{(L-2)}\right), \tag{38}$$

$$= \sum_{n=1}^{N} \log \mathbb{E}_{p\left(f_n^L \,\middle|\, \widetilde{f}_n^{L-1}, \mathbf{u}^{(L)}, \boldsymbol{\Psi}^{(L)}\right)} p\left(\mathbf{z}_n \,\middle|\, f_n^{(L)}; \sigma^2\right) \tag{39}$$

Notice that each step of the approximation is unbiased due to the layer-wise factorization of the joint distribution (Eq. 34). We can obtain a closed form of the last-layer expectation as $\mathbf{z}_n \mid f_n^{(L)}$ is a Gaussian. In the case of using a different distribution, this expectation can be approximated using quadrature (Hensman et al., 2015a).

E. Experimental details

In this section, we present details on implementation and hyperparameters used in our experimental campaign. All experiments were conducted on a server equipped with a Tesla T4 GPU having 16 GB RAM. Throughout all experiments, unless otherwise stated, we impose an isotropic Gaussian prior over the decoder parameters. We use the radial basis function (RBF) kernel with automatic relevance determination (ARD) with marginal variance and independent lengthscales λ_i per feature (MacKay, 1996). We place a lognormal prior with unit variance and means equal to 1 and 0.05 for the lengthscales and variance, respectively. Since the auxiliary data of most of the considered datasets are timestamps, we impose a non-informative uniform prior on the inducing inputs. We observe that this prior works well in our experiments. The lengthscales are initialized to 1, whereas the inducing points are initialized by a k-means algorithm as commonly used in practice (Hensman et al., 2015a). For inference, we use an adaptive version of SGHMC (Springenberg et al., 2016) in which the hyperparameters are automatically tuned during a burn-in phase. The hyperparameters are tuned according to the performance on the validation set.

The random seed for the stochastic inference network is drawn from an isotropic Gaussian distribution, i.e. $q(\boldsymbol{\varepsilon}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. In case the encoder is a multilayer perceptron (MLP), we concatenate the random seeds and the input into a long vector. The dimension of the random seeds is the same as that of the input. If the encoder is a convolutional neural network, we spatially stack the random seeds and the input. We use an Adam optimizer (Kingma & Ba, 2015) with a learning rate of 0.001 for optimizing the encoder network. Unless otherwise specified, we set the default hyperparameters of the number of SGHMC and encoder optimization steps $J = 30$, and $K = 50$, respectively.

E.1. Moving ball experiment

We use the same network architectures as in Jazbec et al. (2021); Pearce (2019). We follow the data generation procedure of Jazbec et al. (2021), in which a squared-exponential GP kernel with a lengthscale $l = 2$ was used. Notice that, unlike Jazbec et al. (2021), we generate a fixed number of 35 videos for training and another 35 videos for testing rather than generating new training videos at each iteration, regardless of the fact that tens of thousands of iterations are performed. This new setting is more realistic and is designed to show the data efficiency of the considered methods. The number of frames in each video is 30. The dimension of each frame is 32×32. Table 6 presents the default hyperparameters used for our SGP-BAE and Gaussian Process Bayesian Autoencoder (GP-BAE) models. For the competing methods, we follow the setups specified in Jazbec et al. (2021).

E.2. Rotated MNIST experiment

For the rotated MNIST experiment, we follow the same setup as in Jazbec et al. (2021) and Casale et al. (2018). In particular, we use the GP kernel proposed by Casale et al. (2018) and a neural network consisting of three convolutional layers followed

by a fully connected layer in the encoder and vice-versa in the decoder. The details of hyperparameters used for our models are presented in Table 7. For the competing methods, we refer to Jazbec et al. (2021).

E.3. Missing imputation experiment

We follow the experimental setting in Ashman et al. (2020), in which squared exponential GP kernels are used. Notice that, to ensure a fair comparison, we handle partially missing data by treating it as zero and feeding it into the inference network (encoder) for SGP-VAE (Ashman et al., 2020) and our SGP-BAE model. This is because it is not trivial to adapt partial inference networks (Ashman et al., 2020) to our stochastic inference network, and we leave this for future work. Table 8 and Table 9 show the default hyperparameters used for our models. For multi-output GP baselines, we refer to Requeima et al. (2019).

Table 6: Parameter settings for the moving ball experiment.

PARAMETER	VALUE
NUM. OF FEEDFORWARD LAYERS IN ENCODER	2
NUM. OF FEEDFORWARD LAYERS IN DECODER	2
WIDTH OF A HIDDEN FEEDFORWARD LAYER	500
DIMENSIONALITY OF LATENT SPACE	2
ACTIVATION FUNCTION	TANH
NUM. OF INDUCING POINTS	10
MINI-BATCH SIZE	FULL
STEP SIZE	0.005
MOMENTUM	0.05
NUM. OF BURN-IN STEPS	1500
NUM. OF SAMPLES	100
THINNING INTERVAL	400

Table 7: Parameter settings for the rotated MNIST experiment.

PARAMETER	VALUE
NUM. OF CONV. LAYERS IN ENCODER	3
NUM. OF CONV. LAYERS IN DECODER	3
NUM. OF FILTERS PER CONV. CHANNEL	8
FILTER SIZE	3×3
NUM. OF FEEDFORWARD LAYERS IN ENCODER	1
NUM. OF FEEDFORWARD LAYERS IN DECODER	1
ACTIVATION FUNCTION	ELU
DIMENSIONALITY OF LATENT SPACE	16
NUM. OF INDUCING POINTS	32
MINI-BATCH SIZE	512
STEP SIZE	0.01
MOMENTUM	0.01
NUM. OF BURN-IN STEPS	1500
NUM. OF SAMPLES	200
THINNING INTERVAL	800

Table 8: Parameter settings for the JURA experiment.

PARAMETER	VALUE
NUM. OF FEEDFORWARD LAYERS IN ENCODER	1
NUM. OF FEEDFORWARD LAYERS IN DECODER	2
WIDTH OF A HIDDEN ENCODER LAYER	20
WIDTH OF A HIDDEN DECODER LAYER	5
DIMENSIONALITY OF LATENT SPACE	2
ACTIVATION FUNCTION	RELU
NUM. OF INDUCING POINTS	128
MINI-BATCH SIZE	100
STEP SIZE	0.002
MOMENTUM	0.05
NUM. OF BURN-IN STEPS	1500
NUM. OF SAMPLES	50
THINNING INTERVAL	180

Table 9: Parameter settings for the EEG experiment.

PARAMETER	VALUE
NUM. OF FEEDFORWARD LAYERS IN ENCODER	1
NUM. OF FEEDFORWARD LAYERS IN DECODER	2
WIDTH OF A HIDDEN ENCODER LAYER	20
WIDTH OF A HIDDEN DECODER LAYER	5
DIMENSIONALITY OF LATENT SPACE	3
ACTIVATION FUNCTION	RELU
NUM. OF INDUCING POINTS	128
MINI-BATCH SIZE	100
STEP SIZE	0.003
MOMENTUM	0.05
NUM. OF BURN-IN STEPS	1500
NUM. OF SAMPLES	50
THINNING INTERVAL	180

F. Additional results

F.1. Ablation study on Bayesian treatments of Autoencoders

There are several approaches to treating autoencoder models in a fully Bayesian manner. In fact, in Appendix E of the seminal paper on VAE, Kingma & Welling (2014) had already considered a fully Bayesian treatment of VAEs by introducing a prior on the decoder. VI is employed to infer the decoder and the latent variables. Daxberger & Hernández-Lobato (2019) suggested employing SGHMC for sampling decoder parameters, but they use the evidence lower bound (ELBO) of the marginal likelihood of VAE as the objective for sampling. This may result in suboptimal approximations. In contrast, our proposed approach entails direct sampling from the posterior of the decoder and latent variables. In order to differentiate our model from these approaches, we name them as Bayesian Variational Autoencoders (BVAEs), following the terminology

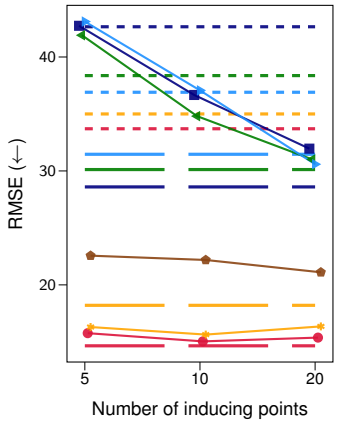
used by Glazunov & Zarras (2022). In particular, we consider the VI (Kingma & Welling, 2014) and SGHMC approaches (Daxberger & Hernández-Lobato, 2019) to treat the decoder of VAEs in a Bayesian manner. These approaches are hereafter referred to as BVAE-VI and BVAE-SGHMC, respectively.

In this section, with the aim of thoroughly disentangling the factors contributing to the superior performance of our proposed models, we present a comprehensive ablation study on these Bayesian treatments of AEs and AE-style models with GP priors. Based on this set of experiments, we identified three key factors that contribute to the improved performance of our proposed models. These factors are: (i) the new amortized SGHMC scheme for inference of the latent variables $\mathbf{Z}$; (ii) treating the decoder in a Bayesian manner and using SGHMC for inference properly; (iii) employing a full Bayesian sparse GP prior on the latent variables and using SGHMC for inference. To evaluate the impact of each factor, we evaluate different modeling and inference choices of the decoder, latent variables, and GP prior. As shown in Fig. 7, the results further demonstrate that our proposal in fact offers the best performance.

Regarding (i), we consider baselines in which the decoder in our models, BAE, GP-BAE, and SGP-BAE, are treated in a non-Bayesian way. In the figure, we name these baselines as BAE-NonBayesDec, GP-BAE-NonBayesDec, SGP-BAE-NonBayesDec, respectively. The results of these new models are slightly worse than the original ones. However, they are still significantly better than variational-based models. Moreover, this also implies the benefit to treat the decoder Bayesian properly (ii).

For (ii), we additionally consider Bayesian VAEs, such as BVAE-VI and BVAE-SGHMC. We find that the Bayesian treatment of the decoder of VAEs is not clearly helpful, even when using SGHMC for the decoder. This is because BVAE-SGHMC still relies on the ELBO as the sampling objective. In contrast, in our models, we sample the decoder directly from the posterior. This is aligned with the recent success of SGHMC on modern Bayesian neural networks (Tran et al., 2022; Izmailov et al., 2021). Moreover, our models jointly sample all parameters at once, which avoids the inefficiency of iterative switching between optimizing the inference network using VI and sampling the decoder.

To verify (iii), we evaluate our model SGP-BAE in the case where the sparse GP is not treated in a fully Bayesian way. The inducing points and kernel parameters are optimized using the objective of Titsias (2009). In Fig. 7, we term this baseline as SGP-BAE-NonBayesGP. We observe that this model performs much worse than our SGP-BAE model.



Model	Decoder φ	Latent variables $\mathbf{Z}$	GP prior
VAE	Non-Bayesian	VI	None
BVAE-VI	VI	VI	None
BAE-SGHMC	SGHMC	VI	None
BAE	SGHMC	SGHMC	None
BAE-NonBayesDec	None	SGHMC	None
GP-VAE	None	VI	Full GP (Fixed)
GP-BVAE-VI	SGHMC	VI	Full GP (Fixed)
GP-BVAE-SGHMC	SGHMC	VI	Full GP (Fixed)
GP-BAE	SGHMC	SGHMC	Full GP (Fixed)
GP-BAE-NonBayesDec	Non-Bayesian	SGHMC	Full GP (Fixed)
SVGP-VAE	Non-Bayesian	VI	Sparse GP (VI)
SVGP-BVAE-VI	VI	VI	Sparse GP (VI)
SVGP-BVAE-SGHMC	SGHMC	VI	Sparse GP (VI)
SGP-BAE	SGHMC	SGHMC	Bayesian sparse GP (SGHMC)
SGP-BAE-NonBayesDec	Non-Bayesian	SGHMC	Bayesian sparse GP (SGHMC)
SGP-BAE-NonBayesGP	SGHMC	SGHMC	Sparse GP (VI)

Figure 7: An ablation study on different Bayesian treatments of AE models and AE-style models with GP priors on the moving ball dataset.

F.2. Latent space visualization

Fig. 8 illustrates a two-dimensional t-SNE (van der Maaten & Hinton, 2008) visualization of latent vectors ($C = 16$) for the rotated MNIST data obtained by our SGP-BAE model. It is evident that the clusters of embeddings are organized in a structured manner according to the angles they represent. Specifically, the embeddings of rotated images are arranged in a continuous sequence from 0 to 2π in a clockwise direction.

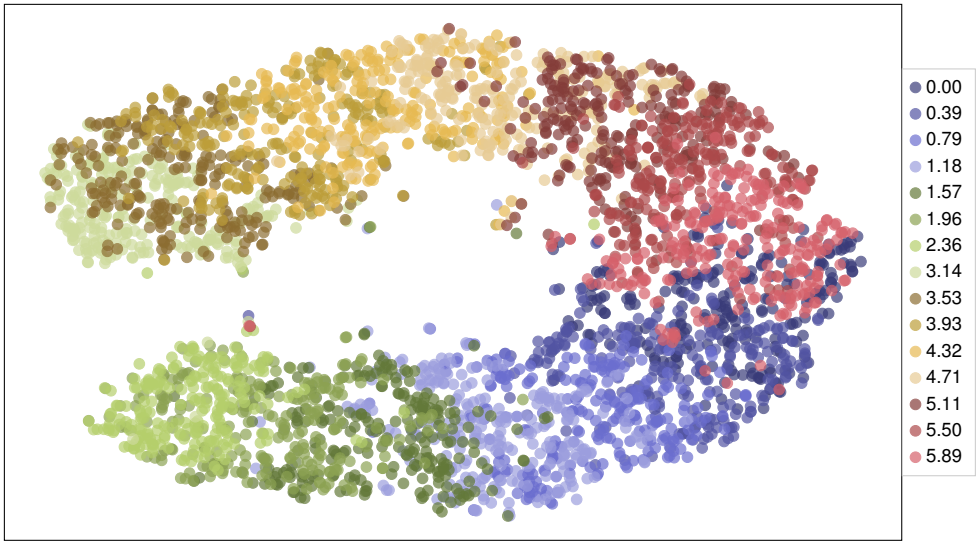


Figure 8: Visualization of t-SNE embeddings (van der Maaten & Hinton, 2008) of SGP-BAE latent vectors on the training data of the rotated MNIST. Each image embedding is colored with respect to its associated angle. Here, we use a perplexity parameter of 50 for t-SNE.

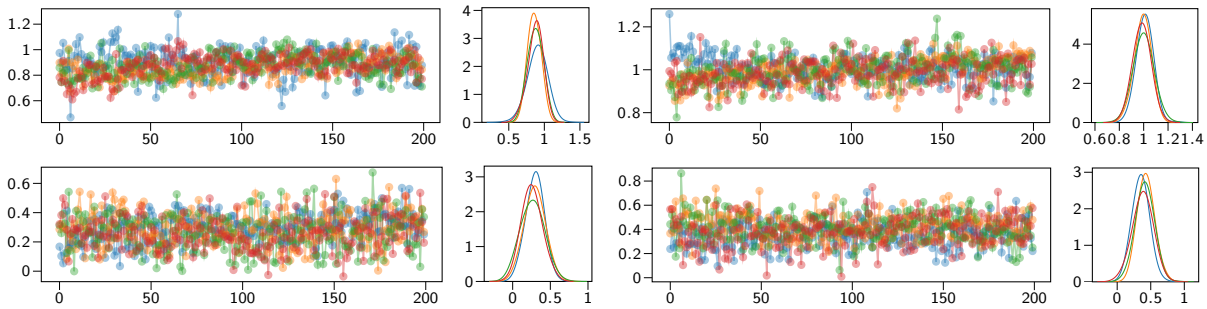


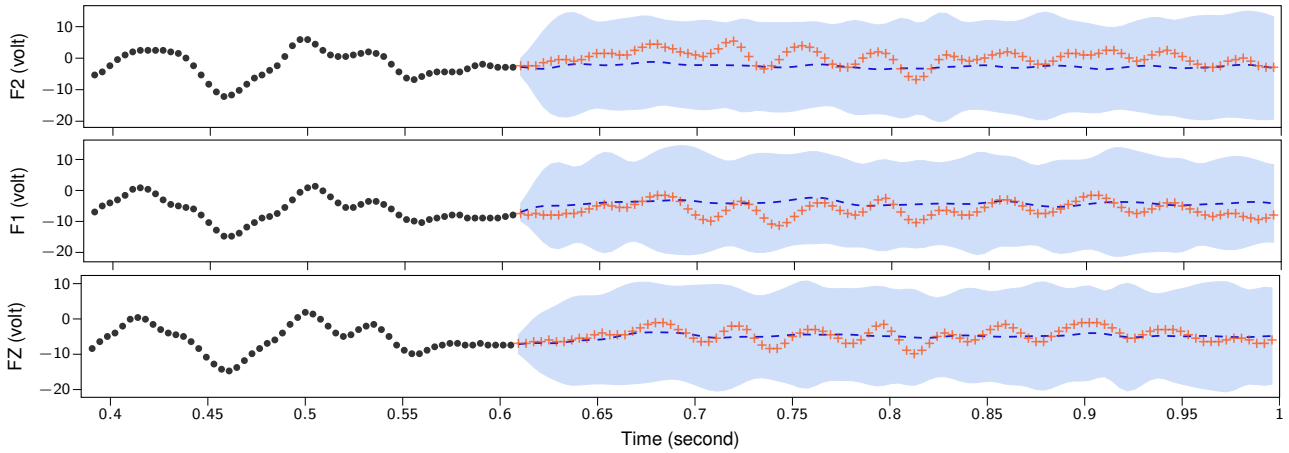
Figure 9: Trace plots for four test points on the rotated MNIST dataset. Here, we simulate 4 chains with 200 samples for each.

F.3. Convergence of SGHMC

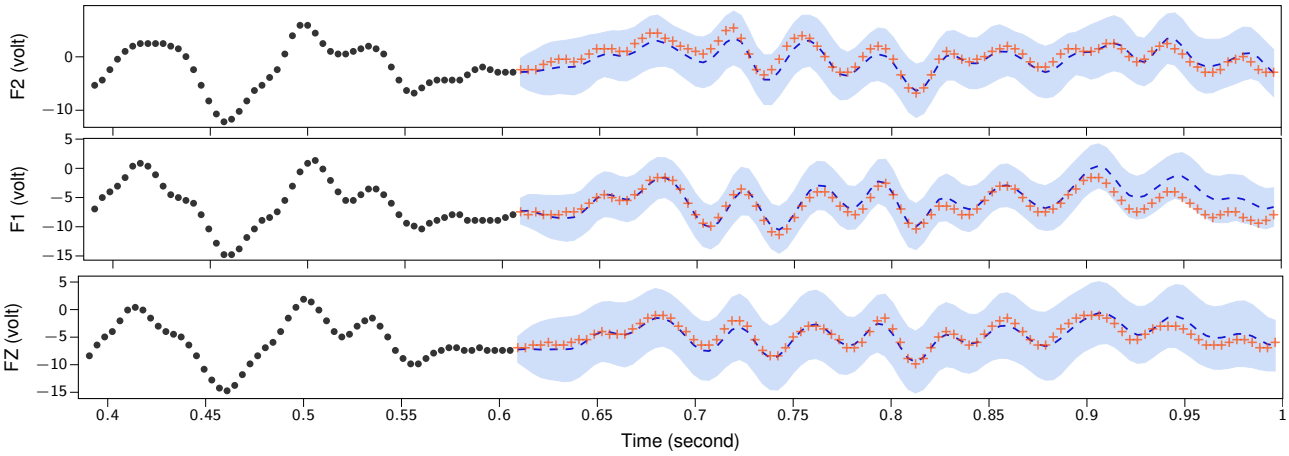
We follow the common practice of using the $\hat{R}$ potential scale-reduction diagnostic (Gelman & Rubin, 1992) to assess the convergence of Markov chain Monte Carlo (MCMC) on Bayesian neural networks (BNNs) (Izmailov et al., 2021; Tran et al., 2022). Given two or more chains, $\hat{R}$ estimates the ratio between the chain variance and the average within-chain variance. For the large-scale MNIST experiment, we compute the $\hat{R}$ statistics on the predictive posterior over four independent chains and obtain a median value of less than 1.1, which suggests good convergence (Izmailov et al., 2021). In addition, we present trace plots of the predictive posterior in Fig. 9, which also support the conclusion of good mixing.

F.4. Visualization of EEG data

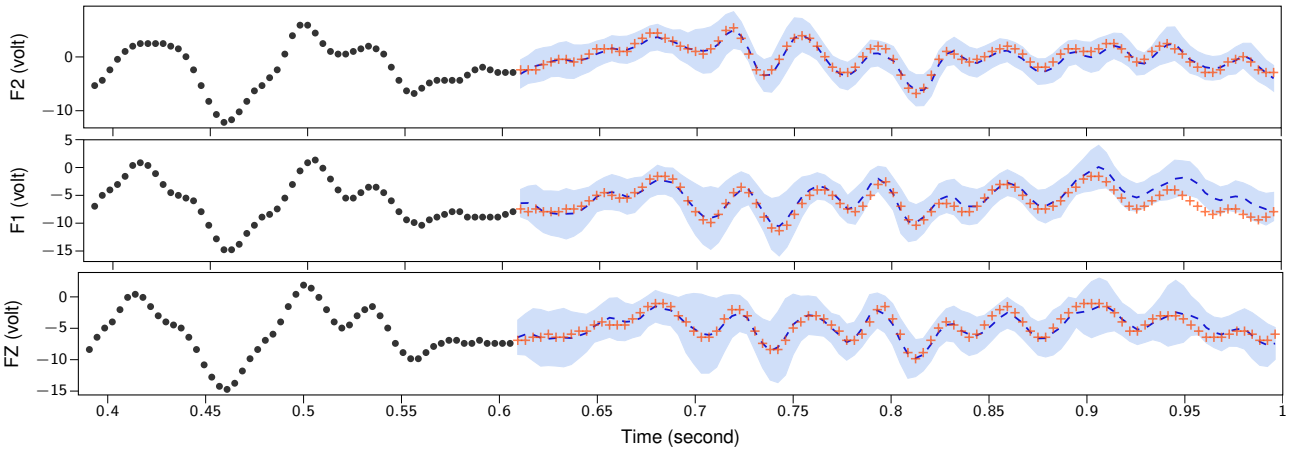
Fig. 10 depicts the predictive mean and uncertainty estimation for the missing values of the EEG dataset.



(a) Independent Gaussian Processes (IGP)



(b) Gaussian Process Autoregressive Model (GPAR)



(c) SGP-BAE

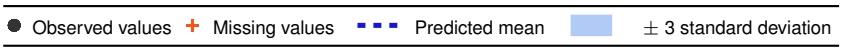


Figure 10: Visualization of predictions for missing data of the EEG dataset. Each panel shows one of the three channels with missing data (orange crosses) and observed data (black points).