Markovian Gaussian Process Variational Autoencoders

Harrison Zhu [1 *] **Carles Balsells-Rodas** [1 *] **Yingzhen Li** [1]

Abstract

Sequential VAEs have been successfully considered for many high-dimensional time series modelling problems, with many variant models relying on discrete-time mechanisms such as recurrent neural networks (RNNs). On the other hand, continuous-time methods have recently gained attraction, especially in the context of irregularly-sampled time series, where they can better handle the data than discrete-time methods. One such class are Gaussian process variational autoencoders (GPVAEs), where the VAE prior is set as a Gaussian process (GP). However, a major limitation of GPVAEs is that it inherits the cubic computational cost as GPs, making it unattractive to practioners. In this work, we leverage the equivalent discrete state space representation of Markovian GPs to enable linear time GPVAE training via Kalman filtering and smoothing. For our model, Markovian GPVAE (MGPVAE), we show on a variety of high-dimensional temporal and spatiotemporal tasks that our method performs favourably compared to existing approaches whilst being computationally highly scalable.

1. Introduction

Modelling multivariate time series data has extensive applications in e.g., video and audio generation (Li & Mandt, 2018; Goel et al., 2022), climate data analysis (Ravuri et al., 2021) and finance (Sims, 1980). Among existing deep generative models for time series, a popular class of model is sequential variational auto-encoders (VAEs) (Chung et al., 2015; Fraccaro et al., 2017; Fortuin et al., 2020), which extend VAEs (Kingma & Welling, 2014) to sequential data. Originally proposed for image generation, a VAE is a latent variable model which encodes data via an **encoder** to a low-dimensional latent space and then decodes via a **decoder** to reconstruct the original data. To extend VAEs to sequential data, the latent space must also include temporal information (it is also technically possible to place temporal dynamics on the decoders (Chen et al., 2017), but for our work we focus on the latent variable dynamics). Sequential VAEs accomplish this by modelling the latent variables as a multivariate time series, where many existing approaches define a state-space model which governs the latent dynamics. These state-space model-based sequential VAE approaches can be classified into two subgroups:

- Discrete-time: The first approach relies on building a **discrete-time state-space model** for the latent variables. The transition distribution is often parameterised by a recurrent neural network (RNN) such as Long Short-Term Memory (LSTM; Hochreiter & Schmidhuber

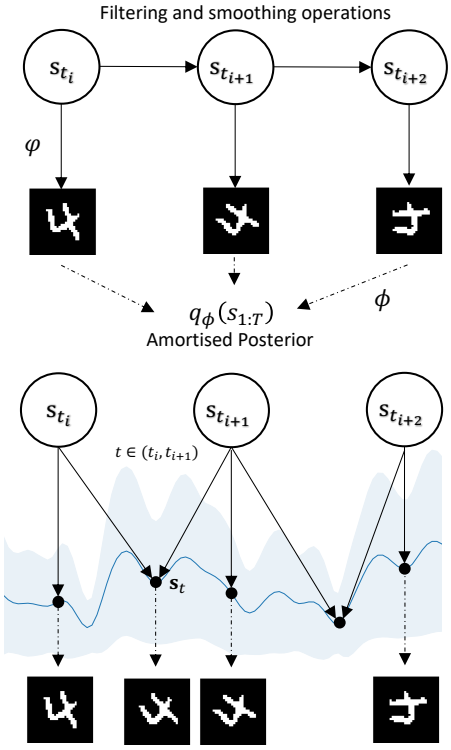

Figure 1: Illustration of the MGPVAE model. (Top) The state posterior $q_\phi(\mathbf{s}_{1:T})$ is parameterised by encoder outputs and computed using filtering and smoothing. (Bottom) At prediction time, posterior predictive distributions can be calculated at any t.

*Equal contribution [1]Imperial College London. Correspondence to: Harrison Zhu <harrisonzhu5080@gmail.com or hbz15@ic.ac.uk>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

(1997)) or Gated Recurrent Units (GRU; Cho et al. (2014)). Notable methods include the Variational Recurrent Neural Network (VRNN; Chung et al. (2015)) and Kalman VAE (KVAE; Fraccaro et al. (2017)). However, these approaches may suffer from training issues, such as vanishing gradients (Pascanu et al., 2013), and may struggle with irregularly-sampled time series data (Rubanova et al., 2019).

- Continuous-time: The second approach involves **continuous-time** representations, where the latent space is modelled using a continuous-time dynamic model. A notable class of such methods are neural differential equations (Chen et al., 2018; Rubanova et al., 2019; Li et al., 2020; Kidger, 2022), which model the latent variables using a system of differential equations, described by its initial conditions, drift and diffusion. As remarked in Li et al. (2020), one can construct a neural stochastic differential equation (neuralSDE) that can be interpreted as an infinite-dimensional noise VAE. However, although these models can flexibly handle irregularly-sampled data, they also require numerical solvers to solve the underlying latent processes, which may cause training difficulties (Park et al., 2021), memory issues (Chen et al., 2018; Li et al., 2020) and slow computation times. Similarly, but combining linear SDEs with Kalman filtering, Continuous Recurrent Units (CRU; Schirmer et al. (2022)) is a RNN that is also able to model continuous data. Finally, in the context of audio generation, S4-related models (Gu et al., 2021; Goel et al., 2022) also rely on continuous state spaces and have been shown to perform strongly.

Another line of continuous-time approaches, related to neuralSDEs, that treat the latent multivariate time series as a random function of time, and model the random function as a tractable stochastic process, are Gaussian Process Variational Autoencoders (GPVAEs; Casale et al. (2018); Pearce (2020); Fortuin et al. (2020); Ashman et al. (2020); Jazbec et al. (2021)) which model the latent variables using Gaussian processes (GPs) (Rasmussen, 2003). As compared with dynamic model-based approaches which focus on modelling the latent variable transitions (reflecting local properties mainly), the GP model for the latent variables describes, in a better way, the global properties of the time series if a suitable stationary kernel is chosen, such as smoothness and periodicity. Therefore GPVAEs may be better suited for e.g., climate time series data which clearly exhibits periodic behaviour. Unfortunately GPVAEs are not directly applicable to long sequences as they suffer from $\mathcal{O}(T^3)$ computational cost, therefore approximations need to be made. Indeed, Ashman et al. (2020); Fortuin et al. (2020); Jazbec et al. (2021) proposed variational approximations based on sparse Gaussian processes (Titsias, 2009; Hensman et al., 2013; 2015) or recognition networks (Fortuin et al., 2020) to improve the scalability of GPVAEs.

In this work we propose Markovian GPVAEs (MGPVAEs) to bridge state-space model-based and stochastic process based approaches of sequential VAEs, aiming to achieve the best in both worlds. Our approach is inspired by the key fact that, when the GP is over time, a large class of GPs can be written as a linear SDE (Särkkä & Solin, 2019), for which there exists exact and unique solutions (Øksendal, 2003). As a result, there exists an equivalent discrete linear state space representation of GPs. Therefore the dynamic model for the latent variables has both discrete and continuous-time representations. This brings the following key advantages to the latent dynamic model of MGPVAE:

- The continuous-time representation allows the incorporation of inductive biases via the GP kernel design (e.g., smoothness, periodic and monotonic trends), to achieve better prediction results and training efficiency. It also enables modelling irregularly sampled time series data.
- The equivalent discrete-time representation, which is linear, enables Kalman filtering and smoothing (Särkkä & Solin, 2019; Adam et al., 2020; Chang et al., 2020; Wilkinson et al., 2020; 2021; Hamelijnck et al., 2021) that computes the posterior distributions in $\mathcal{O}(T)$ time. As the observed data is assumed to come from non-linear transformations of the latent variables, we further apply site-based approximations (Chang et al., 2020) for the non-linear likelihood terms to enable analytic solutions for the filtering and smoothing procedures.

In our experiments, We study much longer datasets ($T \approx 100$) compared to many previous GPVAE and discrete-time works, which are only are of the magnitude of $T \approx 10$. We include a range of datasets that describe different properties of MGPVAE compared to existing approaches:

- We deliver competitive performance compared to many existing methods on corrupt and irregularly-sampled video and robot action data at a fraction of the cost of many existing models.
- We extend our work to spatiotemporal climate data, where none of the discrete-time sequential VAEs are suited for modelling. We show that it outperforms traditional GP and existing sparse GPVAE models in terms of both predictive performance and speed.

2. Background

Consider building generative models for high-dimensional time series (e.g., video data). Here an observed sequence of length T is denoted as $\mathbf{Y}_{t_1} \ldots, \mathbf{Y}_{t_T} \in \mathbb{R}^{D_y}$, where t_i represents the timestamp of the ith observation in the sequence. Note that in general $t_i \neq i$ for irregularly

sampled time series. As the proposed MGPVAE has both discrete state-space model based and stochastic process based formulations, below we introduce these two types of sequential VAEs and the key relevant techniques.

Sequential VAEs with state-space models: Consider $t_i = i$ w.l.o.g., and assume each of the latent states in $\mathbf{Z}_{1:T} = (\mathbf{z}^1_{1:T}, \ldots, \mathbf{z}^L_{1:T}) \in \mathbb{R}^{T\times L}$ has L latent dimensions. Then the generative model is defined as

$$p(\mathbf{Y}_{1:T}, \mathbf{Z}_{1:T}) = p(\mathbf{Z}_{1:T}) \prod_{t=1}^{T} p(\mathbf{Y}_t|\mathbf{Z}_t), \tag{1}$$

where we choose $p(\mathbf{Y}_t|\mathbf{Z}_t) = \mathcal{N}(\mathbf{Y}_t; \varphi(\mathbf{Z}_t), \sigma^2\mathbf{I})$ to be a multivariate Gaussian distribution, and $\varphi : \mathbb{R}^L \to \mathbb{R}^{D_y}$ is decoder network that transforms the latent state to the Gaussian mean. The prior $p(\mathbf{Z}_{1:T})$ is defined by the transition probabilities, e.g., $p(\mathbf{Z}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{Z}_t|\mathbf{Z}_{<t})$. Training is done by maximising the variational lower bound $\mathcal{L}$ (Ranganath et al., 2014):

$$\log p(\mathbf{Y}_{1:T}) \geq \textstyle\sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{Z}_{1:T})}[\log p(\mathbf{Y}_t|\mathbf{Z}_t)] \\ - \mathrm{KL}(q(\mathbf{Z}_{1:T}|\mathbf{Y}_{1:T})||p(\mathbf{Z}_{1:T})) := \mathcal{L}, \tag{2}$$

where $q(\mathbf{Z}_{1:T}|\mathbf{Y}_{1:T})$ is the approximate posterior parameterised by an encoder network. Often this q distribution is defined by mean-field approximation over time, i.e., $q(\mathbf{Z}_{1:T}|\mathbf{Y}_{1:T}) = \prod_{t=1}^{T} q(\mathbf{Z}_t|\mathbf{Y}_t)$, or by using transition probabilities as well, e.g., $q(\mathbf{Z}_{1:T}|\mathbf{Y}_{1:T}) = \prod_{t=1}^{T} q(\mathbf{Z}_t|\mathbf{Z}_{t-1}, \mathbf{Y}_{\leq t})$. Below we also write $q(\mathbf{Z}_{1:T}) = q(\mathbf{Z}_{1:T}|\mathbf{Y}_{1:T})$ to simplify notation.

Gaussian Process Variational Autoencoders: A Gaussian process is a stochastic process denoted by $\mathbf{Z} \sim \mathcal{GP}(0, k)$, where $k : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is the kernel or covariance function that specifies the similarity between two time stamps. This allows us to explicitly enforce inductive biases or global behaviour. For instance, if $\mathbf{Y}_t$ is a periodic system, then we may use a periodic kernel k or a longer initial kernel lengthscale to incorporate this knowledge in the model; if the underlying process is smooth, then a kernel can also be chosen so that it induces smooth functions (Kanagawa et al., 2018).

GPVAEs (Casale et al., 2018; Pearce, 2020; Fortuin et al., 2020; Ashman et al., 2020; Jazbec et al., 2021) define the decoder network φ and the conditional distribution $p(\mathbf{Y}_t|\mathbf{Z}_t)$ in the same way as presented above. However, instead of using transition probabilities, a GPVAE places a multi-output GP prior on the latent variables $\{\mathbf{Z}_t\}$: $\mathbf{Z}_t \sim \mathcal{GP}(0, \mathbf{k})$, where one can choose the kernel of the multi-output GP to be $\mathbf{k} = \mathbf{I} \otimes k$, i.e., the output GPs across dimensions share the same kernel k. However, each dimension may also be induced with separate kernels $k^1, \ldots, k^L$, which we adopt in this work, giving block diagonal kernel matrices.

Again we use the variational lower-bound (Eq. (2) when $t_i = i$) as the training objective, but with a different approximate posterior q. Some examples include GPVAE (Pearce, 2020; Fortuin et al., 2020)

$$q(\mathbf{Z}_{1:T}) = \textstyle\prod_{l=1}^{L} N(\mathbf{Z}^l_{1:T}; \tilde{\mathbf{Y}}^l_{1:T}, \tilde{\mathbf{V}}^l_{1:T}),$$

where $(\tilde{\mathbf{Y}}^l_t, \tilde{\mathbf{V}}^l_t)_{l=1}^L = (\mu^l_\phi(\mathbf{Y}_t), \mathbf{\Sigma}^l_\phi(\mathbf{Y}_t))_{l=1}^L = \phi(\mathbf{Y}_t)$ are outputs of the encoder network ϕ. This corresponds to a mean-field approximation over latent dimensions instead of time. To avoid direct parameterisation of the full covariance matrix $\mathbf{\Sigma}^l_\phi(\mathbf{Y}_{1:T}) \in \mathbb{R}^{T\times T}$ which can be expensive for long sequences, Fortuin et al. (2020) proposed a banded parameterisation of the precision matrix (Blei & Lafferty, 2006; Bamler & Mandt, 2017), reducing both the time and memory complexity to $\mathcal{O}(T)$. However, this choice makes it more difficult to work with irregularly-sampled data and previous works only focused on corrupt video frames.

Another more flexible option is to use sparse GP approximations with inducing points (Jazbec et al., 2021):

$$\begin{aligned} q(\mathbf{Z}_{1:T}) &= \textstyle\prod_{l=1}^{L} p(\mathbf{Z}^l_{1:T}|\mathbf{G}^l_m) q(\mathbf{G}^l_m), \\ q(\mathbf{G}^l_m) &= N(\mathbf{G}^l_m|\mathbf{m}^l_m, \mathbf{A}^l_m), \\ \mathbf{S}^l &= \mathbf{K}^l_{mm} + \mathbf{K}^l_{mT}\mathrm{diag}(\tilde{\mathbf{V}}^l_{1:T})^{-1}\mathbf{K}^l_{Tm}, \\ \mathbf{m}^l_m &= \mathbf{K}^l_{mm}(\mathbf{S}^l)^{-1}\mathbf{K}^l_{mT}\mathrm{diag}(\tilde{\mathbf{V}}^l_{1:T})^{-1}\tilde{\mathbf{Y}}^l_{1:T}, \\ \mathbf{A}^l_m &= \mathbf{K}^l_{mm}(\mathbf{S}^l)^{-1}\mathbf{K}^l_{mm}, \end{aligned}$$

where $p(\mathbf{Z}^l_{1:T}|\mathbf{G}^l_m)$ is the standard multivariate Gaussian conditional distribution and $[\mathbf{K}^l_{mT}]_{ij} := k^l(\mathbf{U}_i, j)$ with $i = 1, \ldots, m$ and $j = 1, \ldots, T$, and $[\mathbf{K}^l_{mm}]_{ij} := k^l(\mathbf{U}_i, \mathbf{U}_j)$ with $i, j = 1, \ldots, m$, for pre-determined inducing time locations $\mathbf{U} = [\mathbf{U}_1, \ldots, \mathbf{U}_m]^\intercal$. However, the time complexity scales with $\mathcal{O}(m^3 + m^2T)$, where in practice to attain good performance $m = \mathcal{O}(\log T)$ (Burt et al., 2019), and therefore the complexity increases massively when dealing with longer time series.

Markovian Gaussian Processes: Interestingly, the banded parameterisation coincides with the structure of the Markovian GP state space $\mathbf{s}_t$. w.l.o.g. suppose $\mathbf{Z}_t$ is one-dimensional in this subsection, then with a conjugate likelihood (e.g. linear Gaussian) $p(\mathbf{Y}_t|\mathbf{Z}_t)$ and a Markovian kernel k (Särkkä & Solin, 2019), we can write the GP regression problem as an Itô SDE of latent dimension d

$$\begin{aligned} \mathrm{d}\mathbf{s}_t &= \mathbf{F}\mathbf{s}_t\mathrm{d}t + \mathbf{L}\mathrm{d}B_t, \quad \mathbf{Z}_t = \mathbf{H}\mathbf{s}_t, \\ \mathbf{Y}_t|\mathbf{Z}_t &\sim p(\mathbf{Y}_t|\mathbf{Z}_t), \end{aligned} \tag{3}$$

where $\mathbf{F} \in \mathbb{R}^{d\times d}, \mathbf{L} \in \mathbb{R}^{d\times e}, \mathbf{H} \in \mathbb{R}^{1\times d}$ are the feedback, noise effect and emission matrices, respectively, and B_t

is an e-dimensional (correlated) Brownian motion with diffusion $\mathbf{Q}_c$. $\mathbf{s}_0 \sim \mathcal{N}(0, \mathbf{P}_\infty)$, where $\mathbf{P}_\infty$ is the stationary state covariance, which satisfies the Lyapunov equation (Solin, 2016). The state is typically the d derivatives $\mathbf{s}_t = (\mathbf{Z}_t, \mathbf{Z}_t^{(1)}, \ldots, \mathbf{Z}_t^{(d-1)})^\intercal$ with the subsequent emission matrix $\mathbf{H} = (1, 0, \ldots, 0)$.

The linear SDE in Eq. (3) admits a unique closed form solution, allowing the recursive updates of $\mathbf{s}_{t_{i+1}}$ given $\mathbf{s}_{t_i}$:

$$\begin{aligned} \mathbf{s}_{t_{i+1}} &= \mathbf{A}_{i,i+1}\mathbf{s}_{t_i} + \mathbf{q}_i, \quad \mathbf{q}_i \sim \mathcal{N}(0, \mathbf{Q}_{i,i+1}), \\ \mathbf{Z}_{t_i} &= \mathbf{H}\mathbf{s}_{t_i}, \quad \mathbf{Y}_{t_i}|\mathbf{Z}_{t_i} \sim p(\mathbf{Y}_{t_i}|\mathbf{Z}_{t_i}) \end{aligned} \tag{4}$$

$$\begin{aligned} &\text{with } \mathbf{A}_{i,i+1} = e^{\Delta_i \mathbf{F}}, \quad \text{where } \Delta_i = t_{i+1} - t_i, \\ &\mathbf{Q}_{i,i+1} = \textstyle\int_{t_0}^{\Delta_i + t_0} e^{(\Delta_i + t_0 - \tau)\mathbf{F}} \mathbf{L}\mathbf{Q}_c\mathbf{L}^\intercal [e^{(\Delta_i + t_0 - \tau)\mathbf{F}}]^\intercal \mathrm{d}\tau. \end{aligned}$$

Note that the $\mathbf{Q}_{i,i+1}$ can be easily obtained in closed form. See Appendix A for a detailed explanation. For conjugate likelihood $p(\mathbf{Y}_t|\mathbf{Z}_t) \equiv p(\mathbf{Y}_t|\mathbf{s}_t)$, we can use the recursive Kalman filtering and smoothing equations (also known as the forward-backward equations) to obtain the posterior distribution (Särkkä & Solin, 2019) $p(\mathbf{s}_t|\mathbf{Y}_{1:T})$ with $\mathcal{O}(Td^3)$ complexity. The corresponding filter prediction, filtering and smoothing equations are:

$$\begin{aligned} p(\mathbf{s}_{t+1}|\mathbf{Y}_{1:t}) &= \textstyle\int p(\mathbf{s}_{t+1}|\mathbf{s}_t) p(\mathbf{s}_t|\mathbf{Y}_{1:t}) \mathrm{d}\mathbf{s}_t, \\ p(\mathbf{s}_t|\mathbf{Y}_{1:t}) &= \tfrac{1}{\ell_t} p(\mathbf{Y}_t|\mathbf{s}_t) p(\mathbf{s}_t|\mathbf{Y}_{1:t-1}), \\ p(\mathbf{s}_t|\mathbf{Y}_{1:T}) &= p(\mathbf{s}_t|\mathbf{Y}_{1:t}) \textstyle\int \frac{p(\mathbf{s}_{t+1}|\mathbf{s}_t) p(\mathbf{s}_{t+1}|\mathbf{Y}_{1:T})}{p(\mathbf{s}_{t+1}|\mathbf{Y}_{1:t})} \mathrm{d}\mathbf{s}_{t+1}, \end{aligned} \tag{5}$$

where $\ell_t = \int p(\mathbf{Y}_t|\mathbf{s}_t) p(\mathbf{s}_t|\mathbf{Y}_{1:t-1}) \mathrm{d}\mathbf{s}_t$ is tractable as the integrand is a product of Gaussians. If $p(\mathbf{Y}_t|\mathbf{s}_t)$ is non-conjugate (e.g. a Poisson distribution), then the posterior cannot be obtained analytically.

The size of d depends on the kernel. For example, the Matern-$3/2$ kernel yields $d = 2$, since the GP sample paths lie in an RKHS that is norm equivalent to a space of functions with 1 derivative (Kanagawa et al., 2018). The periodic or quasi-periodic kernels (Solin & Särkkä, 2014) may yield larger d's. However, in comparison to sparse Gaussian process approximations in Ashman et al. (2020); Jazbec et al. (2021) that have complexity $\mathcal{O}(m^3 + m^2 T)$, where m is the number of inducing points, d does not depend on T (unlike sparse GPs (Burt et al., 2019) that depend on $\mathcal{O}(\log^D T)$, where D is the time variable dimension) and thus does not need to grow as T increases.

3. Markovian Gaussian Process Variational Autoencoders

In this section, we propose Markovian GPVAEs (MGPVAEs) and a corresponding variational inference scheme for model learning.

3.1. Model

Let L be the dimensionality of the latent variables and k^l be the kernel for the lth channel with state dimension d_l. Then we have a total state space dimension of $\sum_{l=1}^L d_l$. Let $\varphi : \mathbb{R}^L \to \mathbb{R}^{D_y}$ be the decoder network. Then the generative model, under the linear SDE form of Markovian GP in the latent space, is (with $\mathbf{Z}_t = (\mathbf{Z}_t^1, \ldots, \mathbf{Z}_t^L)^\intercal$)

$$\begin{aligned} &\mathrm{d}\mathbf{s}_t^l = \mathbf{F}^l \mathbf{s}_t^l \mathrm{d}t + \mathbf{L}^l \mathrm{d}B_t^l, \quad \mathbf{Z}_t^l = \mathbf{H}^l \mathbf{s}_t^l, \quad l = 1, \ldots, L \\ &\mathbf{Y}_t|\mathbf{Z}_t \sim p(\mathbf{Y}_t|\mathbf{Z}_t) \equiv p(\mathbf{Y}_t|\varphi(\mathbf{Z}_t)), \end{aligned} \tag{6}$$

which equivalently becomes the linear discrete state space model with nonlinear likelihood

$$\begin{aligned} &\mathbf{s}_{t_{i+1}}^l = \mathbf{A}_{i,i+1}^l \mathbf{s}_{t_i}^l + \mathbf{q}_i^l, \quad \mathbf{q}_i^l \sim \mathcal{N}(0, \mathbf{Q}_{i,i+1}^l), \\ &\mathbf{Z}_t^l = \mathbf{H}^l \mathbf{s}_t^l, \quad \mathbf{Y}_t|\mathbf{Z}_t \sim p(\mathbf{Y}_t|\mathbf{Z}_t) \equiv p(\mathbf{Y}_t|\varphi(\mathbf{Z}_t)). \end{aligned} \tag{7}$$

Note that the transformation $\mathbf{Z}_t^l = \mathbf{H}^l \mathbf{s}_t^l$ is deterministic, and the stochasticity arises from the $\mathbf{s}_t$ variables.

3.2. Variational inference

Suppose $q(\mathbf{s})$ is the approximate posterior over $\mathbf{s} := \mathbf{s}_{1:T} \in \mathbb{R}^{T \times Ld}$. Then minimising $\mathrm{KL}(q(\mathbf{s})||p(\mathbf{s}|\mathbf{Y}))$ is equivalent to maximising the lower bound $\mathcal{L} := \sum_{t=1}^T \mathbb{E}_{q(\mathbf{s}_t)} \log p(\mathbf{Y}_t|\varphi(\mathbf{Z}_t)) - \mathrm{KL}(q(\mathbf{s})||p(\mathbf{s}))$. Note that since $\mathbf{Z}_t$ is a linear transformation or reparameterisation of $\mathbf{s}_t$, the KL-divergence is between the posterior and prior distributions of $\mathbf{s}_t$. We wish to compute $q(\mathbf{s})$ in linear time using Kalman filtering and smoothing. However, due to the presence of a nonlinear decoder network φ in the likelihood, it is no longer possible to obtain the exact posterior $p(\mathbf{s}|\mathbf{Y})$ due to non-conjugacy. However, if we approximate the likelihood $p(\mathbf{Y}_t|\varphi(\mathbf{Z}_t))$ with Gaussian sites, as is done in Pearce (2020); Ashman et al. (2020); Jazbec et al. (2021); Chang et al. (2020), Kalman filtering and smoothing can be performed as conjugacy is reintroduced in the filtering and smoothing equations.

We propose the Gaussian-site approximation

$$q(\mathbf{s}) \propto p(\mathbf{s}) \textstyle\prod_{l=1}^L \prod_{t=1}^T N(\tilde{\mathbf{Y}}_t^l | \mathbf{H}^l \mathbf{s}_t^l, \tilde{\mathbf{V}}_t^l),$$

where $\tilde{\mathbf{Y}}_t^l \in \mathbb{R}$ and $\tilde{\mathbf{V}}_t^l \in \mathbb{R}$ for $l = 1, \ldots, L$. Instead of optimising $\tilde{\mathbf{Y}}_t$ and $\tilde{\mathbf{V}}_t$ as free-form parameters using conjugate-computation variational inference (Khan & Lin, 2017), we encode them using outputs of an encoder network ϕ i.e. $(\tilde{\mathbf{Y}}_t^l, \tilde{\mathbf{V}}_t^l)_{l=1}^L = \phi(\mathbf{Y}_t)$. In addition, we approximate the potentially high-dimensional data likelihood using a likelihood comprising of low-dimensional state space variables.

Then with straightforward computations,

$$\begin{aligned}&\mathrm{KL}(q(\mathbf{s})||p(\mathbf{s})) = \mathbb{E}_{q(\mathbf{s})}\log\tfrac{q(\mathbf{s})}{p(\mathbf{s})}\\&= \mathbb{E}_{q(\mathbf{s})}\log\tfrac{p(\mathbf{s})\prod_{l=1}^{L}\prod_{t=1}^{T}N(\tilde{\mathbf{Y}}_t^l|\mathbf{H}^l\mathbf{s}_t^l,\tilde{\mathbf{V}}_t^l)}{p(\mathbf{s})\int p(\mathbf{s})\prod_{l=1}^{L}\prod_{t=1}^{T}N(\tilde{\mathbf{Y}}_t^l|\mathbf{H}^l\mathbf{s}_t^l,\tilde{\mathbf{V}}_t^l)\mathbf{ds}}\\&= \mathbb{E}_{q(\mathbf{s})}\textstyle\sum_{l=1}^{l}\sum_{t=1}^{T}\log N(\tilde{\mathbf{Y}}_t^l|\mathbf{H}^l\mathbf{s}_t^l,\tilde{\mathbf{V}}_t^l)\\&- \log\mathbb{E}_{p(\mathbf{s})}\textstyle\prod_{t=1}^{T}\prod_{l=1}^{L}N(\tilde{\mathbf{Y}}_t^l|\mathbf{H}^l\mathbf{s}_t^l,\tilde{\mathbf{V}}_t^l).\end{aligned}$$

The ELBO (2) thus becomes

$$\mathcal{L} = \underbrace{\log\mathbb{E}_{p(\mathbf{s})}\left[\prod_{t=1}^{T}\prod_{l=1}^{L}N(\tilde{\mathbf{Y}}_t^l|\mathbf{H}^l\mathbf{s}_t^l,\tilde{\mathbf{V}}_t^l)\right]}_{\text{E3}}$$

$$+\sum_{t=1}^{T}\mathbb{E}_{q(\mathbf{s}_t)}\left[\underbrace{\log p(\mathbf{Y}_t|\varphi(\mathbf{Z}_t))}_{\text{E2}} - \sum_{l=1}^{L}\underbrace{\log N(\tilde{\mathbf{Y}}_t^l|\mathbf{H}^l\mathbf{s}_t^l,\tilde{\mathbf{V}}_t^l)}_{\text{E1}}\right].$$

(E1) $q(\mathbf{s}_t)$ can be obtained using the Kalman smoothing distributions and can be computed in linear time (see the Kalman filtering and smoothing equations in Appendix 1). The reason is because $q(\mathbf{s})$ is constructed by replacing the likelihood in Eq. (4) with Gaussian sites, making it possible to analytically evaluate the filtering and smoothing equations (5). Therefore using the same calculations as Chang et al. (2020); Hamelijnck et al. (2021), the first term in the ELBO can be computed analytically:

$$\begin{aligned}&\mathbb{E}_{q(\mathbf{s}_t)}[\text{E1}] = -\tfrac{1}{2}\log|2\pi\tilde{\mathbf{V}}_t^l| - \tfrac{1}{2}(\tilde{\mathbf{Y}}_t^l)^\intercal(\tilde{\mathbf{V}}_t^l)^{-1}\tilde{\mathbf{Y}}_t^l\\&+ (\tilde{\mathbf{Y}}_t^l)^\intercal\mathbf{H}^l\mathbf{m}_t^{l,s} - \tfrac{1}{2}[\mathrm{Tr}((\tilde{\mathbf{V}}^l)^{-1}\mathbf{H}^l\mathbf{P}_t^{l,s}(\mathbf{H}^l)^\intercal)]\\&+ (\mathbf{m}_t^{l,s})^\intercal(\mathbf{H}^l)^\intercal\mathbf{H}^l\mathbf{m}_t^{l,s},\end{aligned}$$

where $\mathbf{m}_t^{l,s}$ and $\mathbf{P}_t^{l,s}$ are the smoothing mean and covariances respectively at time t.

(E2) is intractable but we can estimate it using Monte Carlo with K samples $\mathbf{s}_{t,j} \sim q(\mathbf{s}_t)$, and thus samples $\mathbf{Z}_{t,j} = (\mathbf{H}^1\mathbf{s}_{t,j}^1,\dots,\mathbf{H}^L\mathbf{s}_{t,j}^L)^\intercal$:

$$\mathbb{E}_{q(\mathbf{s}_t)}\log p(\mathbf{Y}_t|\varphi(\mathbf{Z}_t)) \approx \tfrac{1}{K}\textstyle\sum_{j=1}^{K}\log p(\mathbf{Y}_t|\varphi(\mathbf{Z}_{t,j})).$$

(E3) is the log partition function of $q(\mathbf{s})$, which is also the log marginal likelihood of the approximate model $\sum_{l=1}^{L}\log p(\tilde{\mathbf{Y}}^l)$. Note that the latent channels are independent of each other, allowing us to sum over the log marginal likelihood over each channel. We can further decompose each term of the sum into

$$\begin{aligned}\log p(\tilde{\mathbf{Y}}^l) &= \log p(\tilde{\mathbf{Y}}_1^l)\textstyle\prod_{t=2}^{T}p(\tilde{\mathbf{Y}}_t^l|\tilde{\mathbf{Y}}_{1:t-1}^l)\\&= \textstyle\sum_{t=1}^{T}\log\mathbb{E}_{p(\mathbf{s}_t^l|\tilde{\mathbf{Y}}_{1:t-1}^l)}N(\tilde{\mathbf{Y}}_t^l;\mathbf{H}^l\mathbf{s}_t^l,\tilde{\mathbf{V}}_t^l),\end{aligned}$$

where $p(\mathbf{s}_t^l|\tilde{\mathbf{Y}}_{1:t-1}^l)$ is the predictive filter distribution obtained with Kalman filtering. Fortunately, $\log p(\tilde{\mathbf{Y}}^l)$ can be computed during the filtering stage (see Algorithm 1 for a full breakdown of Kalman filtering and smoothing). In addition, a graphical representation of the Markovian GPVAE is in Figure 1.

3.3. Spatiotemporal Modelling

Spatiotemporal modelling is an important task with many real world applications (Cressie, 2015). Traditional methods such as kriging, or Gaussian process regression, incurs cubic computational costs and are even more costly and difficult when multiple variables need to be modelled jointly. GPVAEs may ameliorate this issue by effectively simplifying the task via an encoder-decoder model and has been proven to be effective in Ashman et al. (2020). Following section 4.2 of Hamelijnck et al. (2021), given a separable spatiotemporal kernel $k(r,t,r',t') = k_r(r,r')k_t(t,t')$, it is straightforward to extend MGPVAE to model spatiotemporal data, which will be make it a highly scalable spatiotemporal model. We consider the model:

$$\mathbf{Z}(r,t) \sim \mathcal{GP}(0,k), \mathbf{Y}(r,t)|\mathbf{Z}(r,t) \sim p(\mathbf{Y}(r,t)|\varphi(\mathbf{Z}(r,t))),$$

where for classical kriging (Cressie, 2015), φ is the identity map and $\mathbf{Z}(r,t)$ is of the same dimensionality as $\mathbf{Y}(r,t)$.

For convenience of notation, we avoid introducing subscripts l and only write down 1 latent dimension with k. Suppose we have N_s spatial coordinates observed over time, denoted by the spatial matrix $\mathbf{R} \in \mathbb{R}^{N_r \times D_x}$, it is possible to rewrite the GP regression model by stacking the states for each spatial location on top of each other to get:

$$\begin{aligned}&\mathbf{s}_{t_{i+1}} = \mathbf{A}_{i,i+1}\mathbf{s}_{t_i} + \mathbf{q}_i, \quad \mathbf{q}_i \sim \mathcal{N}(0,\mathbf{Q}_{i,i+1}),\\&\mathbf{Y}_t|\mathbf{Z}_t \sim p(\mathbf{Y}_t|\varphi(\mathbf{Z}_t)),\end{aligned} \tag{8}$$

where $\mathbf{s}_t = [\mathbf{s}_t(r_1),\dots,\mathbf{s}_t(r_{N_s})]^\intercal$, $\mathbf{Z}_t = [\mathbf{L}_{\mathbf{RR}}^r \otimes \mathbf{H}^t]\mathbf{s}_{t_i}$ and $\mathbf{A}_{i,i+1} = \mathbf{I}_{N_r} \otimes \mathbf{A}_{i,i+1}^t$, $\mathbf{Q}_{i,i+1} = \mathbf{I}_{N_r} \otimes \mathbf{Q}_{i,i+1}^t$, $\mathbf{K}_{\mathbf{RR}}^r = \mathbf{L}_{\mathbf{RR}}^r(\mathbf{L}_{\mathbf{RR}}^r)^\intercal$, where superscripts t and r indicate the temporal state space and spatial kernel matrices respectively. The graphical model for MGPVAE is shown in Figure 2, demonstrating how the states for each spatial location are independently filtered and smoothed over time, and then spatially mixed by the emission matrix. Lastly, we approximate the likelihood with a mean-field amortised approximation $\prod_{t=1}^{T}N(\tilde{\mathbf{Y}}_t|[\mathbf{L}_{\mathbf{RR}}^r \otimes \mathbf{H}^t]\mathbf{s}_t,\tilde{\mathbf{V}}_t)$, where $\tilde{\mathbf{Y}}_t, \tilde{\mathbf{V}}_t \in \mathbb{R}^{N_x}$. See Appendix A.3 for further details.

3.4. Computational Complexity and Storage

Computational Complexity: For simplicity let us assume that each channel has the same kernel but is modelled independently. Then the computational complexity of MGPVAE is $\mathcal{O}(Ld^3T)$. For GPVAE, it is also $\mathcal{O}(Ld^3T)$; for SVGPVAE, $\mathcal{O}(L(Tm^2+m^3))$ with m inducing points. For KVAE, VRNN and CRU, the complexity is $\mathcal{O}(LT)$, but there may be large big-O constants due to the RNN network sizes. For neuralODEs and neuralSDEs, the complexity is linear with respect to the number of discretisation steps, which can potentially be much larger than T.

For spatiotemporal modelling, the computational complexity for MGPVAE will be $\mathcal{O}(Ld^3TN_r^3)$ in this case,

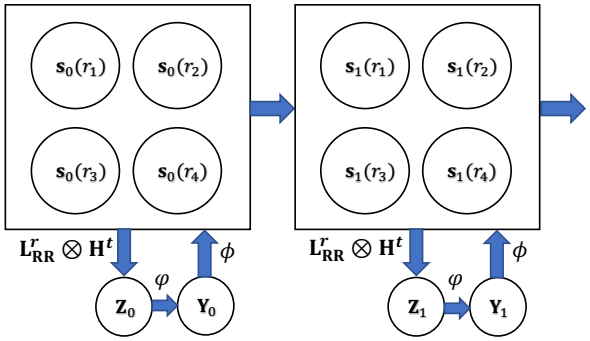

Figure 2: Graphical model of the separable spatiotemporal MGPVAE model. The temporal dynamics of the states $\mathbf{s}_t(r)$ at each location r are independently handled. At each time step t, these states are spatially mixed to produce $\mathbf{Z}_t$, which is then transformed by a non-linear mapping to $\mathbf{Y}_t$.

but can be further lowered to $\mathcal{O}(Ld^3T(N_r m^2 + m^3))$ if we sparsify the spatial domain by using M spatial inducing points. For SVGPVAE, it is $\mathcal{O}(L(TN_r m^2 + m^3))$ with m inducing points over space and time.

Storage: The storage requirements for MGPVAE and SVGPVAE are $\mathcal{O}(Td^2)$ and $\mathcal{O}(Tm + m^2)$ respectively. For larger T, m needs to be larger and hence $m \gg d^2$ in many cases (e.g. $d^2 = 4$ for the Matern-$3/2$ kernel).

4. Related Work

GPVAEs have been explored for time series modelling in Fortuin et al. (2020); Ashman et al. (2020). In our work, we experiment with the state-of-the-art GPVAEs of Fortuin et al. (2020) and SVGPVAE (Jazbec et al., 2021), and explore a wider variety of tasks. Unlike Fortuin et al. (2020), MGPVAE is capable of tackling both corrupt and missing frames imputation tasks, as well as spatiotemporal modelling tasks, with great scalability. Compared to sparse GPVAEs (Ashman et al., 2020; Jazbec et al., 2021), MGPVAE does not require inducing points.

NeuralODEs (Chen et al., 2018; Rubanova et al., 2019) and neuralSDEs (Li et al., 2020) are also continuous-time models that can tackle the same tasks as MGPVAE. However, they depend heavily on the time discretisation, how well the initial conditions are learned and the expressiveness of the drift and diffusion functions. In our work, we experiment with neuralODEs, which have previously been used for similar missing frames imputation tasks, and find that it is the slowest model whithout achieving good predictive performance.

Many discrete-time and continuous-time models, such as VRNN (Chung et al., 2015), KVAE (Fraccaro et al., 2017) and latentODE (Rubanova et al., 2019), are only designed to model temporal, but not spatiotemporal, data. On the other hand, classical multioutput GPs can only handle lower-dimensional spatiotemporal datasets as there cannot be any dimensionality reduction to a latent space, whereas GPVAE enables the encoder-decoder networks to learn meaningful low-dimensional representations for high-dimensional data. SVGPVAE is able to handle spatiotemporal data, but has the disadvantage of being less efficient than MGPVAE due to the use of inducing points over space and time jointly.

Chang et al. (2020); Hamelijnck et al. (2021) considered modelling non-conjugate likelihoods with Gaussian approximations, which would allow for Kalman filtering and smoothing operations. We adopt this strategy to allow flexible decoder choices with non-linear mapping, where a key difference is that the encoder helps us construct a low-dimensional approximation to the likelihood function, which allows us to work with Kalman filtering and smoothing in the lower-dimensional latent space.

5. Experiments

We present 3 sets of experiments: rotating MNIST, Mujoco action data and spatiotemporal data modelling. We benchmark MGPVAE against a variety of continuous and discrete time models, such as GPVAE (Fortuin et al., 2020), SVGPVAE (Jazbec et al., 2021), KVAE (Fraccaro et al., 2017), VRNN (Chung et al., 2015), LatentODE (Rubanova et al., 2019), CRU (Schirmer et al. (2022); only report RMSE as it was not originally conceived as a generative model) and sparse variational multioutput GP (MOGP). We evaluate the performances using both test negative log-likelihood (NLL) and root mean squared error (RMSE). We implemented each model across different libraries (JAX, PyTorch and TensorFlow) due to varying suitabilities and tried our best to optimise each implementation for fairness of comparison. All wall-clock time computations are done on NVIDIA RTX-3090 GPUs with 24576MiB RAM. See further experimental details and results in Appendix B.

5.1. Rotating MNIST

In this experiment, we produce sequences of MNIST frames in which the digits are rotated with a periodic length of 50, over $T = 100$ frames. We tackle 2 imputation tasks for: corrupted frames where the frame pixels are randomly set to 0, and missing frames where frames are randomly dropped out of each sequence. Each task has 4000 /1000 train/test sequences respectively. The underlying dynamics are simple (rotation) which may favour models with stronger inductive biases, such as GPVAE and MGPVAE. To test this, for both models, we use Matern-$3/2$ kernels with lengthscales initialised at 40 (fixed for GPVAE according to Fortuin et al. (2020)).

Corrupt frames imputation: This is a task that highly suits standard RNN-based models such as VRNN and

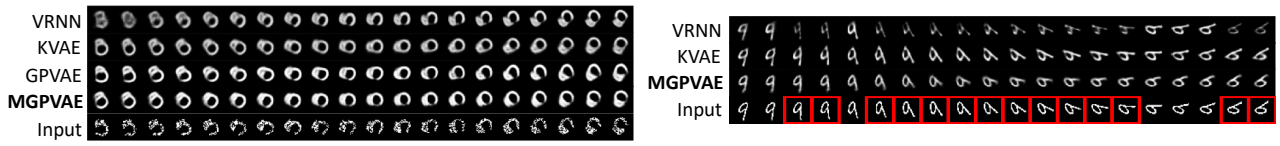

Figure 3: (Left) Corrupt frames imputation results for an unseen sequence of 5's. (Right) Missing frames imputation results for an unseen sequence of 9's. Missing frames are red frames.

Table 1: Test NLL and RMSE for both the corrupt (Cor) and missing frames (Mis) imputation tasks.

Model	NLL-Cor (↓)	RMSE-Cor (↓)	Time-Cor (s/epoch ↓)	NLL-Mis (↓)	RMSE-Mis (↓)	Time-Mis (s/epoch ↓)
VRNN	9898 ± 162.0	0.1768 ± 0.001563	63.51	16240 ± 2090	0.1796 ± 0.008002	103.6
KVAE	12500 ± 83.13	0.2025 ± 0.0006077	139.2	10730 ± 1232	0.1582 ± 0.008688	149.0
GPVAE	9026 ± 48.70	$\mathbf{0.1340 \pm 0.0004529}$	**48.93**	NA	NA	NA
MGPVAE	$\mathbf{8556 \pm 69.66}$	$\mathbf{0.1468 \pm 0.0006738}$	**50.45**	$\mathbf{8925 \pm 53.40}$	$\mathbf{0.1508 \pm 0.0005190}$	**59.43**

KVAE, since the frames are observed at regular time steps. Even so, we see from Table 1 that both GPVAE and MGPVAE perform significantly better than VRNN and KVAE in both NLL and RMSE, validating our hypothesis of inductive biases helping the learning dynamics. Furthermore, we observe that the RMSE for MGPVAE is worse than GPVAE, although it has a slightly better NLL. However, the left panel of Figure 3 shows that the images generated do not visually differ significantly, which implies that the model performance is comparable.

Missing frames imputation: Rubanova et al. (2019) showed that RNN-based models fail in imputing irregularly sampled time-series, as they struggle to correctly update the hidden states at time steps of unobserved frames. This is confirmed by our results in Table 1: MGPVAE outperforms both VRNN and KVAE in terms of NLL and RMSE.[1] In the right panel of Figure 3, we illustrate the posterior mean imputations, and again VRNN fails. These results are expected since VRNN implements filtering (only includes past observations), while MGPVAE and KVAE include a smoothing step (includes both past and future observations).

5.2. Mujoco Action Data

The Mujoco dataset is a physical simulation dataset generated using the Deepmind Control Suite (Tunyasuvunakool et al., 2020). We obtained the Hopper generation code from Rubanova et al. (2019), which outputs 14 dimensions sequences, and for all models we use 15-dimensional latent dimensions (according to Rubanova et al. (2019)). We modify the task so that we only train on the observed time steps, whereas in Rubanova et al. (2019) the models have access to data at all the time steps. This makes the task harder as the model has less information to work with during training. We have 2 settings (1) 1280/400 train-test split with length $T = 100$ and (2) 320/100 for length $T = 1000$. Compared to rotating MNIST, the underlying nonlinear dynamics are more complex, and each dimension can behave differently. For both SVGPVAE and MGPVAE, we use Matern-$^3/_2$ kernels.

Results: We see from Table 2 that the performance of VRNN, KVAE, latentODE and CRU are significantly worse than GP-based models, and overall both SVGPVAE and MGPVAE achieve the best NLL and RMSE. This is because missing data imputation is a difficult task for the discrete-time RNN-based models. On the other hand, latentODE significantly underperforms, possibly due to the ELBO being computed only over the observed time steps, making it more difficult to fit the model than the original task in Rubanova et al. (2019). Figure 4 confirms the results; indeed the non-GP models struggle to simultaneously fit the data and estimate uncertainty well. Additional results can be found in Appendix B.2.

In terms of time complexity, VRNN, KVAE, latentODE and CRU are comparable to each other as shown in Figure 5. The time and memory complexities for SVGPVAE is dependent on the number of inducing points (see section 3.4) and there is a trade-off between model expressiveness and inducing points. For longer sequences, such as for $T = 1000$, we would have to choose more inducing points to gain comparable performance to MGPVAE. We see that with only 20 inducing points, although the wall-clock time is faster than MGPVAE, SVGPVAE-20 underperforms MGPVAE; with 40 inducing points, although the performance and time complexities are comparable to MGPVAE, it also has maxed-out the memory on our GPU. In comparison, MGPVAE does not require any inducing points and is the most time-efficient model.

5.3. Spatiotemporal Climate Data

We obtained climate data, including temperature and precipitation, from ERA5 using Google Earth Engine API (Gorelick et al., 2017). The task is to condition on observed data $\{\mathbf{Y}(r,t)\}_{r,t}$ (temperature and air pressure), and predict on unknown spatial locations $\{\mathbf{Y}(r_*,t)\}_{r_*,t}$ for all time steps. Here we focus on GP-based models

[1] We omit GPVAE here as the implementation by Fortuin et al. (2020) cannot efficiently handle missing frames in batches. See Appendix B.

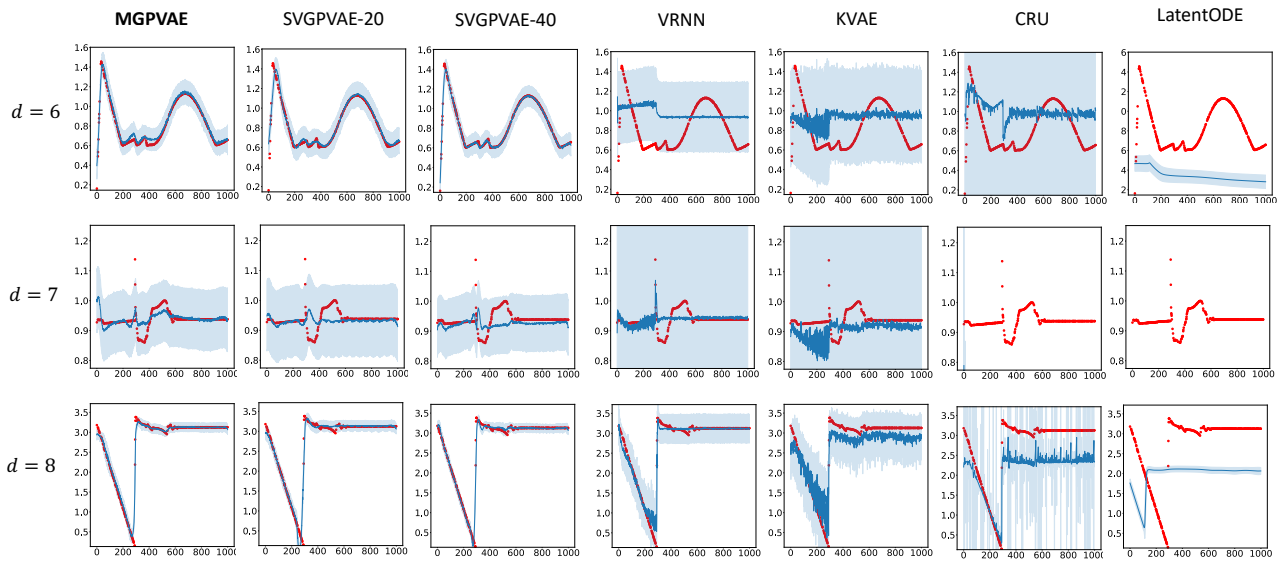

Figure 4: 95% posterior credible intervals for unseen mujoco sequences in its 6,7 and 8th dimensions with $T = 1000$. The red dots show observed data. Note that some predictions are not showing as they fall outside the limits.

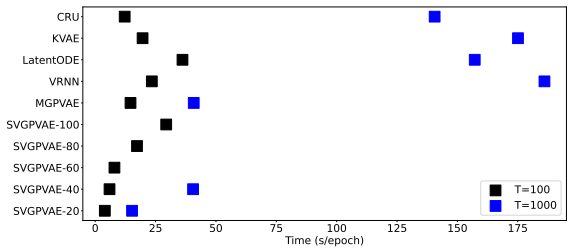

Figure 5: Wallclock times for each model in the Mujoco experiment.

Table 2: Imputation results for the Mujoco tasks.

Model	NLL ($T = 100$) (↓)	RMSE ($T = 100$) (↓)	NLL ($T = 1000$) (↓)	RMSE ($T = 1000$) (↓)
CRU	-	0.1343 ± 0.009169	-	0.1353 ± 0.008574
VRNN	-385.2 ± 25.59	0.1774 ± 0.002499	-2877 ± 450.8	0.1844 ± 0.004053
KVAE	-8.353 ± 25.6	0.1828 ± 0.004587	-262.4 ± 1141	0.1761 ± 0.01751
LatentODE	124 ± 11.99	0.06599 ± 0.0007146	240.6 ± 38.62	0.07749 ± 0.001119
SVGPVAE-20	$\mathbf{-2438 \pm 111}$	0.02841 ± 0.003288	-18020 ± 282.6	0.0538 ± 0.0007901
SVGPVAE-40	$\mathbf{-2468 \pm 106.4}$	$\mathbf{0.02566 \pm 0.002945}$	-21290 ± 136.8	0.04237 ± 0.000295
SVGPVAE-60	-2290 ± 53.69	0.03014 ± 0.001579	-	-
SVGPVAE-80	-2312 ± 67.76	0.0287 ± 0.001866	-	-
MGPVAE	-2292 ± 18.02	0.03068 ± 0.0006991	$\mathbf{-21610 \pm 233.7}$	$\mathbf{0.04156 \pm 0.0007136}$

since they, unlike RNN-based models, can flexibly handle spatiotemporal data by combining spatial and temporal kernels to efficiently model correlated structures. In particular, for GPVAE models we use a separable kernel $k(r, t, r', t) = k_r(r, r')k_t(t, t')$ for each latent channel and the spatial kernel k_r is shared across channels. We also consider Gaussian process prediction which is also known as kriging in spatial statistics (Cressie, 2015). Sparse approximation is needed for scalability for the MOGP baseline due to computational feasibility. We emphasise that the dimensionality of this problem, which is 8, is high relative to traditional spatiotemporal modelling problems (Cressie, 2015).

Results: We report the corresponding quantitative results in Table 3 and visualise the prediction results at an unseen spatial location in Figure 6. We observe that MOGP underfits and overestimates the uncertainty, which is expected as traditional GP regression models struggle to fit a complicated multi-dimensional time series with non-warped GPs (without decoder network). In comparison, the encoder-decoder networks allow GPVAEs to learn simpler dynamics, resulting in better performance. The conclusions are similar when we fix the time steps and plot the corresponding posterior mean over space in Figure 7, where GPVAE models better predict the spatial patterns.

SVGPVAE underperforms MGPVAE even when pushing the number of inducing points to the memory limit of our hardware (500). Interestingly, SVGPVAE underestimates the uncertainty more as we increase the number of latent channels to 8, as can also be seen from the large negative log-likelihood per dim (NLPD) values. It is unclear why this occurs, though this could be related to an imbalance in the regularisation effect with the KL divergence. We note that MGPVAE does not suffer from the same issue.

SVGPVAE results can be improved if having well-placed and sufficiently-many inducing points, but this also means SVGPVAE is memory inefficient when compared with MGPVAE. For the spatiotemporal modelling task, a significantly larger number of inducing points is required and thus further illustrates the drawbacks of SVGPVAE for spatiotemporal modelling. Similarly for computational time, MGPVAE is faster than the other models with more inducing points, but slower than the ones with less of inducing points (which also perform worse).

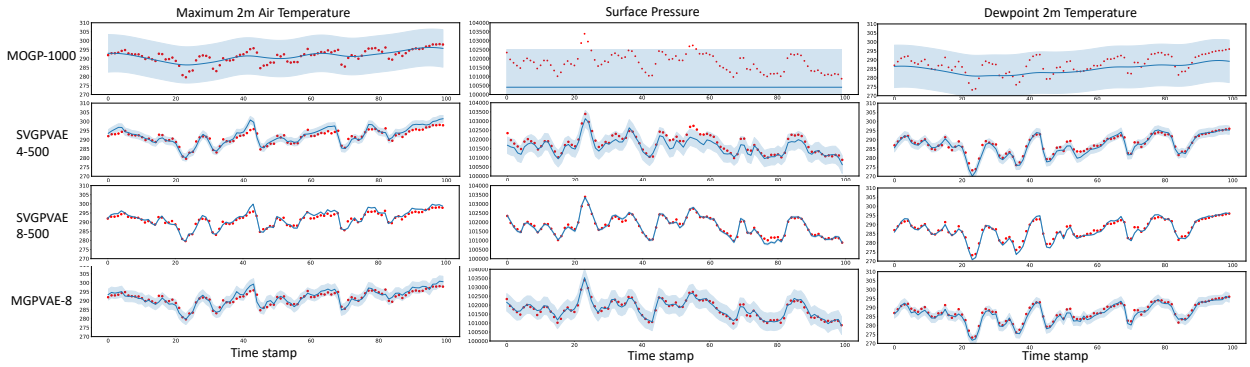

Figure 6: 95% posterior credible intervals for climate variables at an unseen spatial location.

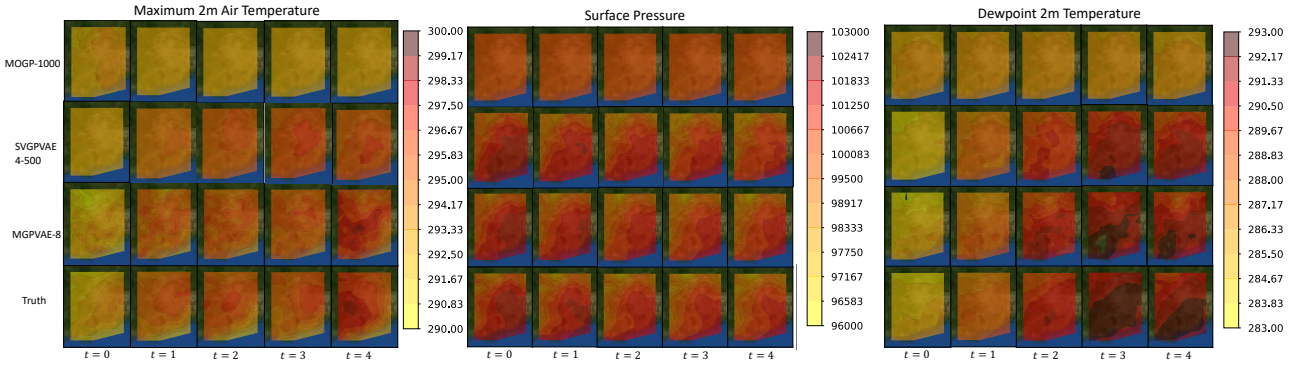

Figure 7: Posterior means over space for climate variables over 5 time steps.

Table 3: ERA5 prediction results.

Model	NLPD (↓)	RMSE (↓)	Time (s/epoch ↓)
MOGP-100	10.38 ± 0.1783	1119 ± 21.29	0.1220
MOGP-1000	10.38 ± 0.1784	1119 ± 21.13	0.6001
SVGPVAE-4-100	16.40 ± 1.531	434.04 ± 11.40	0.05987
SVGPVAE-4-500	8.222 ± 0.9483	388.8 ± 9.347	0.7046
SVGPVAE-8-100	1376 ± 473.0	392.2 ± 18.08	0.08675
SVGPVAE-8-500	2613 ± 540.6	$\mathbf{313.9 \pm 21.42}$	1.344
MGPVAE-4	2.454 ± 0.1149	402.1 ± 15.24	0.45876
MGPVAE-8	$\mathbf{-0.3070 \pm 0.1021}$	$\mathbf{352.7 \pm 24.86}$	0.5128

6. Conclusion and Discussion

We propose MGPVAE, a GPVAE model using Markovian Gaussian processes that uses Kalman filtering and smoothing with linear time complexity. This is achieved by approximating the non-Gaussian likelihoods using Gaussian sites. Compared to Fortuin et al. (2020), which also achieves linear time complexity by using an approximate covariance structure, our model leaves the original Gaussian process covariance structure intact, and additionally work with both irregularly-sampled time series and spatiotemporal data. Experiments on video, Mujoco action and climate modelling tasks show that our method is both competitive and scalable, compared to modern discrete and continuous-time models. Future work can explore the use of parallel filtering (Särkkä & García-Fernández, 2020), other nonlinear filtering approaches (Kamthe et al., 2022) and forecasting applications.

Limitations: MGPVAE requires the use of kernels that admit a Markovian decomposition, and therefore kernels such that the squared exponential kernel will not be permissible (Särkkä & Solin, 2019) without additional approximations. However, we argue that most commonly used kernels are indeed Markovian, which should be sufficient for most applications. In addition, we need to use Gaussian site approximations for the likelihood, in order to allow for Kalman filtering and smoothing. This may lower the approximation accuracy, though in our experiments we see that MGPVAE still performs strongly.

7. Acknowledgements

We would like to especially thank Wenlin Chen for his valuable help with code and experiments during the revision period, especially implementing a more optimised version of SVGPVAE with functorch (Horace He, 2021) compared to the original TensorFlow implementation of Jazbec et al. (2021). HZ was supported by the EPSRC Centre for Doctoral Training in Modern Statistics and Statistical Machine Learning (EP/S023151/1) and the Department of Mathematics of Imperial College London. HZ was supported by Cervest Limited. We would also like to thank Andy Thomas for his endless support with using the NVIDIA4, Forrest and NVIDIA6 GPU Compute Servers.

References

Adam, V., Eleftheriadis, S., Artemev, A., Durrande, N., and Hensman, J. Doubly sparse variational Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pp. 2874–2884. PMLR, 2020.

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.

Bamler, R. and Mandt, S. Dynamic word embeddings. In *International conference on Machine learning*, pp. 380–389. PMLR, 2017.

Blei, D. M. and Lafferty, J. D. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pp. 113–120, 2006.

Burt, D., Rasmussen, C. E., and Van Der Wilk, M. Rates of convergence for sparse variational gaussian process regression. In *International Conference on Machine Learning*, pp. 862–871. PMLR, 2019.

Casale, F. P., Dalca, A., Saglietti, L., Listgarten, J., and Fusi, N. Gaussian process prior variational autoencoders. *Advances in neural information processing systems*, 31, 2018.

Chang, P. E., Wilkinson, W. J., Khan, M. E., and Solin, A. Fast variational learning in state-space Gaussian process models. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6. IEEE, 2020.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. Variational lossy autoencoder. *ICLR*, 2017.

Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-4012. URL https://aclanthology.org/W14-4012.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. A recurrent latent variable model for sequential data. *Advances in neural information processing systems*, 28, 2015.

Cressie, N. *Statistics for spatial data*. John Wiley & Sons, 2015.

Developers, O. Objax, 2020. URL https://github.com/google/objax.

Evans, L. C. An introduction to stochastic differential equations version 1.2. *Lecture Notes, UC Berkeley*, 2006.

Fortuin, V., Baranchuk, D., Rätsch, G., and Mandt, S. Gp-vae: Deep probabilistic time series imputation. In *International conference on artificial intelligence and statistics*, pp. 1651–1661. PMLR, 2020.

Fraccaro, M., Kamronn, S., Paquet, U., and Winther, O. A disentangled recognition and nonlinear dynamics model for unsupervised learning. *Advances in neural information processing systems*, 30, 2017.

Goel, K., Gu, A., Donahue, C., and Ré, C. It's raw! audio generation with state-space models. In *International Conference on Machine Learning*, pp. 7616–7633. PMLR, 2022.

Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., and Moore, R. Google earth engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment*, 2017. doi: 10.1016/j.rse.2017.06.031. URL https://doi.org/10.1016/j.rse.2017.06.031.

Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Hamelijnck, O., Wilkinson, W., Loppi, N., Solin, A., and Damoulas, T. Spatio-temporal variational Gaussian processes. *Advances in Neural Information Processing Systems*, 34, 2021.

Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. *UAI*, 2013.

Hensman, J., Matthews, A., and Ghahramani, Z. Scalable variational gaussian process classification. In *Artificial Intelligence and Statistics*, pp. 351–360. PMLR, 2015.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Horace He, R. Z. functorch: Jax-like composable function transforms for pytorch. https://github.com/pytorch/functorch, 2021.

Jazbec, M., Ashman, M., Fortuin, V., Pearce, M., Mandt, S., and Rätsch, G. Scalable gaussian process variational autoencoders. In *International Conference on Artificial Intelligence and Statistics*, pp. 3511–3519. PMLR, 2021.

Kamthe, S., Takao, S., Mohamed, S., and Deisenroth, M. Iterative state estimation in non-linear dynamical systems using approximate expectation propagation. *Transactions on Machine Learning Research*, 2022.

Kanagawa, M., Hennig, P., Sejdinovic, D., and Sriperumbudur, B. K. Gaussian Processes and Kernel Methods: A Review on Connections and Equivalences. *arXiv:1807.02582 [cs, stat]*, July 2018. URL http://arxiv.org/abs/1807.02582. arXiv: 1807.02582.

Khan, M. and Lin, W. Conjugate-computation variational inference: Converting variational inference in non-conjugate models to inferences in conjugate models. In *Artificial Intelligence and Statistics*, pp. 878–887. PMLR, 2017.

Kidger, P. On neural differential equations. *PhD Thesis*, 2022.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, May 2014. URL http://arxiv.org/abs/1312.6114. arXiv: 1312.6114.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pp. 3744–3753. PMLR, 2019.

Li, X., Wong, T.-K. L., Chen, R. T., and Duvenaud, D. Scalable gradients for stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, pp. 3870–3882. PMLR, 2020.

Li, Y. and Mandt, S. Disentangled sequential autoencoder. *ICML*, 2018.

Lin, Z. and Yin, F. Towards flexibility and interpretability of gaussian process state-space model. *arXiv preprint arXiv:2301.08843*, 2023.

Maroñas, J. and Hernández-Lobato, D. Efficient transformed gaussian processes for non-stationary dependent multi-class classification. *arXiv preprint arXiv:2205.15008*, 2022.

Matthews, A. G. d. G., Van Der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., León-Villagrá, P., Ghahramani, Z., and Hensman, J. Gpflow: A gaussian process library using tensorflow. *J. Mach. Learn. Res.*, 18(40):1–6, 2017.

Park, S., Kim, K., Lee, J., Choo, J., Lee, J., Kim, S., and Choi, E. Vid-ode: Continuous-time video generation with neural ordinary differential equation. *arXiv preprint arXiv:2010.08188*, pp. online, 2021.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pp. 1310–1318. PMLR, 2013.

Pearce, M. The gaussian process prior vae for interpretable latent dynamics from pixels. In *Symposium on Advances in Approximate Bayesian Inference*, pp. 1–12. PMLR, 2020.

Ranganath, R., Gerrish, S., and Blei, D. Black box variational inference. In *Artificial intelligence and statistics*, pp. 814–822. PMLR, 2014.

Rasmussen, C. E. Gaussian processes in machine learning. In *Summer school on machine learning*, pp. 63–71. Springer, 2003.

Ravuri, S., Lenc, K., Willson, M., Kangin, D., Lam, R., Mirowski, P., Fitzsimons, M., Athanassiadou, M., Kashem, S., Madge, S., et al. Skilful precipitation nowcasting using deep generative models of radar. *Nature*, 597(7878):672–677, 2021.

Rubanova, Y., Chen, R. T., and Duvenaud, D. K. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

Särkkä, S. and García-Fernández, Á. F. Temporal parallelization of bayesian smoothers. *IEEE Transactions on Automatic Control*, 66(1):299–306, 2020.

Särkkä, S. and Solin, A. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

Schirmer, M., Eltayeb, M., Lessmann, S., and Rudolph, M. Modeling irregular time series with continuous recurrent units. In *International Conference on Machine Learning*, pp. 19388–19405. PMLR, 2022.

Sims, C. A. Macroeconomics and reality. *Econometrica: journal of the Econometric Society*, pp. 1–48, 1980.

Solin, A. Stochastic differential equation methods for spatio-temporal Gaussian process regression. 2016. PhD Thesis: Aalto University.

Solin, A. and Särkkä, S. Explicit link between periodic covariance functions and state space models. In *AISTATS*, 2014.

Taylor, S. J. and Letham, B. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.

Tebbutt, W., Solin, A., and Turner, R. E. Combining pseudo-point and state space approximations for sum-separable Gaussian processes. In *Uncertainty in Artificial Intelligence*, pp. 1607–1617. PMLR, 2021.

Titsias, M. Variational learning of inducing variables in sparse gaussian processes. In *Artificial intelligence and statistics*, pp. 567–574. PMLR, 2009.

Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., Heess, N., and Tassa, Y. dmcontrol: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638. doi: https://doi.org/10.1016/j.simpa.2020.100022. URL https://www.sciencedirect.com/science/article/pii/S2665963820300099.

Wilkinson, W., Chang, P., Andersen, M., and Solin, A. State Space Expectation Propagation: Efficient Inference Schemes for Temporal Gaussian Processes. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10270–10281. PMLR, July 2020. URL https://proceedings.mlr.press/v119/wilkinson20a.html.

Wilkinson, W. J., Solin, A., and Adam, V. Sparse Algorithms for Markovian Gaussian Processes. *arXiv:2103.10710 [cs, stat]*, June 2021. URL http://arxiv.org/abs/2103.10710. arXiv: 2103.10710.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Øksendal, B. Stochastic differential equations. In *Stochastic differential equations*, pp. 65–84. Springer, 2003.

A. Markovian Gaussian Process Background

In this section, we provide a rigorous treatment of the derivation of the state space Markovian Gaussian process equations as previously presented in Särkkä & Solin (2019) with stochastic differential equation technicalities from Øksendal (2003); Evans (2006).

A.1. Stochastic Differential Equations

Definition A.1 (Univariate Brownian Motion; (Øksendal, 2003; Evans, 2006))**.** Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space. $B : [0, T] \times \Omega \to \mathbb{R}$ is a univariate Brownian motion if:

1. $B_0 = 0$, almost surely.
2. $t \mapsto B(t, \cdot)$ is continuous almost surely.
3. For $t_4 > t_3 \geq t_2 > t_1$, $B_{t_4} - B_{t_3} \perp\!\!\!\perp B_{t_2} - B_{t_1}$.
4. For any $\delta > 0$, $B_{t+\delta} - B_t \sim \mathcal{N}(0, \delta q)$, where q the correlation factor.

In a similar manner, we may extend univariate Brownian motions into correlated multi-dimensional forms.

Definition A.2 (Multi-dimensional Brownian Motion)**.** B_t is an e-dimensional (correlated) Brownian motion: each B^k, for $k = 1, \ldots, e$, is a univariate Brownian motion. We assume that each B^k is correlated, resulting in $B_{t+\delta} - B_t \sim \mathcal{N}(0, \delta \mathbf{Q}_c)$, where $\mathbf{Q}_c \in \mathbb{R}^{e \times e}$ is the spectral density matrix.

Define $\mathcal{F}_t$ to be the σ-algebra generated by $B_s(\cdot)$ for $s \leq t$. Intuitively, this is the 'history of information' up to time t created by B_s for $s \leq t$. Therefore for a stochastic process driven by B_s up to time t, we would like it to be measurable at all times t, which motivates the next definition of measurability for stochastic processes:

Definition A.3. Let $\{\mathcal{F}_t\}_{t \geq 0}$ be an increasing family of σ-algebras generated by B_t i.e. for $s < t$, $\mathcal{F}_s \subset \mathcal{F}_t$. Then $v(t, \omega) : [0, \infty) \times \Omega \to \mathbb{R}^e$ is $\mathcal{F}_t$-adapted if $v(t, \omega)$ is measurable in $\mathcal{F}_t$ for all $t \geq 0$.

With the correlated formulation, we can rederive Itô isometry using the same proof as in the uncorrelated case (Øksendal, 2003; Evans, 2006). We use the definition of the multi-dimensional Itô integral in Definition 3.3.1 Øksendal (2003). Let $\mathbb{E}$ be the expectation with respect to $\mathbb{P}$.

Definition A.4 (Multi-dimensional Itô Integral (Øksendal, 2003))**.** Let $\mathcal{V}^{d \times e}(S, T)$, for $d \in \mathbb{N}$ and $S < T$, be a set of matrix-valued stochastic processes $v(t, \omega) \in \mathbb{R}^{d \times e}$ where each entry $v_{ij}(t, \omega)$ satisfies:

- $(t, \omega) \to v_{ij}(t, \omega)$ is $\mathcal{B}(\mathbb{R}) \times \mathcal{F}$-measurable, where $\mathcal{B}(\mathbb{R})$ is the Borel σ-algebra on $[0, \infty)$.
- $v_{ij}(\tau, \omega) \in L^2([S, T])$ in expectation i.e. $\mathbb{E}[\int_S^T v_{ij}^2(\tau, \cdot) \mathrm{d}\tau]$.
- Let W_t be a univariate Brownian motion. There exists an increasing family of σ-algebras $\{\mathcal{H}_t\}_{t \geq 0}$ such that (1) W_t is a martingale with respect to $\mathcal{H}_t$ and (2) $v_{ij}(t, \omega)$ is $\mathcal{H}_t$-adapted.

Then, we can define the multi-dimensional Itô integral as follows: For all $v \in \mathcal{V}^{d \times e}(S, T)$,

$$\int_S^T v(\tau, \omega) \mathrm{d}B_\tau = \int_S^T \begin{pmatrix} v_{11}(\tau, \omega) & \cdots & v_{1e}(\tau, \omega) \\ \vdots & \ddots & \vdots \\ v_{d1}(\tau, \omega) & \cdots & v_{de}(\tau, \omega) \end{pmatrix} \begin{pmatrix} \mathrm{d}B_\tau^1 \\ \vdots \\ \mathrm{d}B_\tau^e \end{pmatrix},$$

where $\left[\int_S^T v(\tau, \omega) \mathrm{d}B_\tau\right]_i = \sum_{j=1}^e \int_S^T v_{ij}(\tau, \omega) dB_\tau^j$.

Proposition A.5 (Multi-dimensional Itô Isometry)**.** *For $\mathbf{F}, \mathbf{G} \in \mathcal{V}^{d \times e}$ (as defined in Øksendal (2003)) and $S < T$, then*

$$\mathbb{E}[\int_S^T \mathbf{F}(\tau, \cdot) dB_\tau][\int_S^T \mathbf{G}(\tau, \cdot) dB_\tau]^\intercal = \mathbb{E} \int_S^T \mathbf{F}(\tau, \cdot) \mathbf{Q}_c \mathbf{G}(\tau, \cdot)^\intercal d\tau.$$

Furthermore, if $\mathbf{F}$ and $\mathbf{G}$ are deterministic, then for $t_1, t_2 \geq t_0$,

$$\mathbb{E}[\int_{t_0}^{t_1} \mathbf{F}(\tau) dB_\tau][\int_{t_0}^{t_2} \mathbf{G}(\tau) dB_\tau]^\intercal = \mathbb{E} \int_{t_0}^{\min(t_1, t_2)} \mathbf{F}(\tau) \mathbf{Q}_c \mathbf{G}(\tau)^\intercal d\tau.$$

Proof. Let $\{e_i\}_{i=1}^e$ be the canonical basis in $\mathbb{R}^e$. We first show that ik-th component of $\left[\mathbb{E}[\int_S^T \mathbf{F}(\tau,\cdot)\mathrm{d}B_\tau][\int_S^T \mathbf{G}(\tau,\cdot)\mathrm{d}B_\tau]^\intercal\right] \in \mathbb{R}^{d\times d}$, for $S<T$, is equal to

$$\begin{aligned}\int_S^T \mathbf{F}_i^\intercal(\tau,\cdot)\mathbf{Q}_c[\mathbf{G}(\tau,\cdot)]_k^\intercal \mathrm{d}\tau &= \mathbb{E}\int_S^T e_i^\intercal \mathbf{F}^\intercal(\tau,\cdot)\mathbf{Q}_c[\mathbf{G}(\tau,\cdot)]_k^\intercal \mathrm{d}\tau \\ &= e_i^\intercal\left[\mathbb{E}\int_S^T \mathbf{F}^\intercal(\tau,\cdot)\mathbf{Q}_c\mathbf{G}(\tau,\cdot)^\intercal \mathrm{d}\tau\right]e_k \\ &= \left[\mathbb{E}\int_S^T \mathbf{F}^\intercal(\tau,\cdot)\mathbf{Q}_c\mathbf{G}(\tau,\cdot)^\intercal \mathrm{d}\tau\right]_{ik},\end{aligned}$$

which would complete the proof. Indeed, the ik-th component is equal to

$$\begin{aligned}\mathbb{E}\left[\sum_{j=1}^e \int_S^T \mathbf{F}_{ij}(\tau,\cdot)\mathrm{d}B_\tau^j\right]\left[\sum_{l=1}^e \int_S^T \mathbf{G}_{kl}(\tau,\cdot)\mathrm{d}B_\tau^l\right] &= \left[\sum_{j,l=1}^e \mathbb{E}\int_S^T \mathbf{F}_{ij}(\tau,\cdot)\mathrm{d}B_\tau^j \int_S^T \mathbf{G}_{kl}(\tau,\cdot)\mathrm{d}B_\tau^l\right] \\ &\overset{\text{1D Itô Isometry}}{=} \left[\sum_{j,l=1}^e \mathbb{E}\int_S^T \mathbf{F}_{ij}(\tau,\cdot)[\mathbf{Q}_c]_{jl}\mathbf{G}_{kl}(\tau,\cdot)\mathrm{d}\tau\right] \\ &= \sum_{j,l=1}^e \mathbb{E}\int_S^T \mathbf{F}_{ij}(\tau,\cdot)[\mathbf{Q}_c]_{jl}[\mathbf{G}^\intercal(\tau,\cdot)]_{lk}\mathrm{d}\tau \\ &= \mathbb{E}\int_S^T e_i^\intercal \mathbf{F}(\tau,\cdot)\mathbf{Q}_c[\mathbf{G}^\intercal(\tau,\cdot)]e_k\mathrm{d}\tau \\ &= \left[\mathbb{E}\int_S^T \mathbf{F}(\tau,\cdot)\mathbf{Q}_c[\mathbf{G}^\intercal(\tau,\cdot)]\mathrm{d}\tau\right]_{ik},\end{aligned}$$

as required. Next, if $\mathbf{F}$ and $\mathbf{G}$ are deterministic and suppose that $t_1<t_2$ without loss of generality, then

$$\begin{aligned}\mathbb{E}[\int_{t_0}^{t_1} \mathbf{F}(\tau)\mathrm{d}B_\tau][\int_{t_0}^{t_2} \mathbf{G}(\tau)\mathrm{d}B_\tau]^\intercal &= \mathbb{E}[\int_{t_0}^{t_1} \mathbf{F}(\tau)\mathrm{d}B_\tau][\int_{t_0}^{t_1} \mathbf{G}(\tau)\mathrm{d}B_\tau + \cancel{\int_{t_1}^{t_2} \mathbf{G}(\tau)\mathrm{d}B_\tau}]^\intercal \\ &= \mathbb{E}[\int_{t_0}^{\min(t_1,t_2)} \mathbf{F}(\tau)\mathrm{d}B_\tau][\int_{t_0}^{\min(t_1,t_2)} \mathbf{G}(\tau)\mathrm{d}B_\tau]^\intercal \\ &= \mathbb{E}\int_{t_0}^{\min(t_1,t_2)} \mathbf{F}(\tau)\mathbf{Q}_c\mathbf{G}^\intercal(\tau)\mathrm{d}B_\tau,\end{aligned}$$

where for the first line we used property (iii) of Definition A.1 of Brownian motions in multi-dimensions. □

A.2. Markovian Gaussian Processes

A Markovian Gaussian process $f\sim\mathcal{GP}(0,k)$ can be written with an SDE of latent dimension d

$$\mathrm{d}\mathbf{s}(t) = \mathbf{F}\mathbf{s}(t)\mathrm{d}t + \mathbf{L}\mathrm{d}B_t, \quad f(x) = \mathbf{H}\mathbf{s}(t), \tag{9}$$

where $\mathbf{F}\in\mathbb{R}^{d\times d}, \mathbf{L}\in\mathbb{R}^{d\times e}, \mathbf{H}\in\mathbb{R}^{1\times d}$ are the feedback, noise effect and emission matrices, and B_t is an e-dimensional (correlated) Brownian motion with spectral density matrix $\mathbf{Q}_c$. Suppose that $\mathbf{s}(t_0)\sim\mathcal{N}(\mathbf{m}_0,\mathbf{P}_0)$ with the stationary state mean and covariance. Note that we assume that $s(t_0)$ independent of $\mathcal{F}_{t_0}^+$, the σ-algebra generated by B_t-B_s for $t\geq s\geq t_0$, in order to invoke existence and uniquess of SDE solutions (Existence and Uniqueness Theorem, page 90, Evans (2006)). Thus the linear SDE equation 3 admits a unique closed form solution:

$$\mathbf{s}(t) = e^{(t-t_0)\mathbf{F}}\mathbf{s}(t_0) + \int_{t_0}^t e^{(t-\tau)\mathbf{F}}\mathbf{L}\mathrm{d}B_\tau.$$

We have

$$m(t) = \mathbb{E}[\mathbf{s}(t)] = e^{(t-t_0)\mathbf{F}}\mathbf{m}_0,$$

since for any $f \in \mathcal{V}^{d\times e}$, $\mathbb{E}[\int_S^T f(\tau)\mathrm{d}B_\tau] = 0$ (Øksendal, 2003; Evans, 2006). To calculate the covariance, we have

$$\begin{aligned}
\kappa(t,t') &= \mathbb{E}[\mathbf{s}(t) - m(t))(\mathbf{s}(t') - m(t'))^\intercal] \\
&= e^{(t-t_0)\mathbf{F}}\mathbb{E}[(\mathbf{s}(t_0) - \mathbf{m}_0)(\mathbf{s}(t_0) - \mathbf{m}_0)^\intercal][e^{(t'-t_0)\mathbf{F}}]^\intercal \\
&+ \mathbb{E}[\int_{t_0}^t e^{(t-\tau)\mathbf{F}}\mathrm{d}B_\tau][\int_{t_0}^{t'} e^{(t'-\tau)\mathbf{F}}\mathrm{d}B_{\tau'}]^\intercal \\
&+ \cancel{\mathbb{E}\left[e^{(t-t_0)\mathbf{F}}\mathbf{L}(\mathbf{s}(t_0) - \mathbf{m}_0)(\int_{t_0}^{t'} e^{(t'-\tau)\mathbf{F}}\mathbf{L}\mathrm{d}B_\tau)^\intercal\right]} \quad \text{(Independence)} \\
&+ \cancel{\mathbb{E}\left[(\int_{t_0}^t e^{(t-\tau)\mathbf{F}}\mathbf{L}\mathrm{d}B_\tau)(\mathbf{s}(t_0) - \mathbf{m}_0)^\intercal\mathbf{L}^\intercal[e^{(t-t_0)\mathbf{F}}]^\intercal\right]} \quad \text{(Independence)} \\
&= e^{(t-t_0)\mathbf{F}}\mathbf{P}_0[e^{(t'-t_0)\mathbf{F}}]^\intercal + \int_{t_0}^{\min(t,t')} e^{(t-\tau)\mathbf{F}}\mathbf{L}\mathbf{Q}_c\mathbf{L}^\intercal[e^{(t'-\tau)\mathbf{F}}]^\intercal\mathrm{d}\tau \quad \text{(Itô Isometry; Proposition A.5)},
\end{aligned}$$

where we note that $\int_{t_0}^t e^{(t-\tau)\mathbf{F}}\mathbf{L}\mathrm{d}B_\tau$ and $\int_{t_0}^{t'} e^{(t'-\tau)\mathbf{F}}\mathbf{L}\mathrm{d}B_\tau$ are $\mathcal{F}_t$-measurable for $\mathcal{F}_t \subset \mathcal{F}_{t_0}^+$, and thus independent to $\mathbf{s}(t_0)$.

Therefore for all $t \in \{t_0, \ldots, t_T\}$, if we solve for each t_i with initial time t_{i-1}, then $s(t) \sim \mathcal{N}(m(t), k(t,t))$ with

$$\begin{aligned}
m(t_{i+1}) &= e^{(t_{i+1}-t_i)\mathbf{F}}\mathbf{m}_0 \\
k(t_{i+1}, t_{i+1}) &= e^{(t_{i+1}-t_i)\mathbf{F}}\mathbf{P}_0[e^{(t_{i+1}-t_i)\mathbf{F}}]^\intercal + \int_{t_i}^{t_{i+1}} e^{(t_{i+1}-\tau)\mathbf{F}}\mathbf{L}\mathbf{Q}_c\mathbf{L}^\intercal[e^{(t_{i+1}-\tau)\mathbf{F}}]^\intercal\mathrm{d}\tau.
\end{aligned}$$

Denote $\Delta_{t_{i+1}} = t_{i+1} - t_i$, $\mathbf{A}_{i,i+1} = e^{(t_{i+1}-t_i)\mathbf{F}} = e^{\Delta_{t_{i+1}}\mathbf{F}}$ and $\mathbf{Q}_{i,i+1} = \int_{t_0}^{\Delta_{t_{i+1}}+t_0} e^{(\Delta_{t_{i+1}}+t_0-\tau)\mathbf{F}}\mathbf{L}\mathbf{Q}_c\mathbf{L}^\intercal[e^{(\Delta_{t_{i+1}}+t_0-\tau)\mathbf{F}}]^\intercal\mathrm{d}\tau$. Then we can derive the recursive equations

$$\begin{aligned}
\mathbf{s}(t_{i+1}) &= \mathbf{A}_{i,i+1}\mathbf{s}(t_i) + \int_{t_i}^{t_{i+1}} e^{(t_{i+1}-\tau)\mathbf{F}}\mathbf{L}\mathrm{d}B_\tau \\
&= \mathbf{A}_{i,i+1}\mathbf{s}(t_i) + \int_{t_0}^{\Delta_{t_{i+1}}+t_0} e^{\Delta_{t_{i+1}}+t_0-\tau}\mathbf{L}\mathrm{d}B_\tau \\
&= \mathbf{A}_{i,i+1}\mathbf{s}(t_i) + \mathbf{q}_i,
\end{aligned}$$

where $\mathbf{q}_i \sim \mathcal{N}(0, \mathbf{Q}_{i,i+1})$.

Types of Kernels: A large number of kernels do allow for the Markovian property to be satisfied. A full list of them could be found in Solin (2016); Särkkä & Solin (2019). Here, we list 2 common kernels:

- Matern-$3/2$: The kernel is $k(t,t') = \sigma^2(1 + \frac{\sqrt{3}d}{\rho})\exp(-\frac{\sqrt{3}d}{\rho})$, where $d = |t-t'|$. With $\lambda = \frac{\sqrt{3}}{\ell}$:

$$\mathbf{F} = \begin{pmatrix} 0 & 1 \\ -\lambda^2 & -2\lambda \end{pmatrix}, \quad \mathbf{m}_0 = \mathbf{0}, \quad \mathbf{P}_0 = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2\lambda^2 \end{pmatrix}, \quad \mathbf{Q}_c = \frac{12\sqrt{3}\sigma^2}{\ell^2}, \quad \mathbf{L} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \mathbf{H} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

- Matern-$5/2$: The kernel is $k(t,t') = \sigma^2(1 + \frac{\sqrt{5}d}{\rho} + \frac{5d^2}{3\rho^2})\exp(-\frac{\sqrt{5}d}{\rho})$, where $d = |t-t'|$. With $\lambda = \frac{\sqrt{5}}{\ell}$ and $\kappa = \frac{5\sigma^2}{3\ell^2}$:

$$\mathbf{F} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\lambda^3 & -3\lambda^2 & -3\lambda \end{pmatrix}, \quad \mathbf{m}_0 = \mathbf{0}, \quad \mathbf{P}_0 = \begin{pmatrix} \sigma^2 & 0 & \kappa \\ 0 & \kappa & 0 \\ -\kappa & 0 & \frac{25\sigma^2}{\ell^4} \end{pmatrix}, \quad \mathbf{Q}_c = \frac{400\sqrt{5}\sigma^2}{3\ell^5}, \quad \mathbf{L} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad \mathbf{H} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

Kalman Filtering and Smoothing: The full Kalman filtering and smoothing algorithm is delineated in Algorithm 1.

A.3. Further details on spatiotemporal MGPVAE

To perform prediction at arbitrary spatiotemporal locations (r_*, t_*), we can use a conditional independence property proven in Tebbutt et al. (2021) for separable spatiotemporal Markovian GPs that yields the predictive distribution

$$\begin{aligned}
p(\mathbf{Z}(r_*, t_*)|\mathbf{Y}) &\approx q(\mathbf{Z}(r_*, t_*)) \\
&:= \int p(\mathbf{Z}(r_*, t_*)|\mathbf{s}(\mathbf{R}, t_*))q(\mathbf{s}(\mathbf{R}, t_*)|\mathbf{s}(\mathbf{R}, t_{1:T}))\mathrm{d}\mathbf{s}(\mathbf{R}, t_*),
\end{aligned} \tag{10}$$

Algorithm 1 Kalman Filtering and Smoothing

Inputs: Variational parameters $\tilde{\mathbf{Y}}_{1:T}, \tilde{\mathbf{V}}_{1:T}$. Initial conditions $\mathbf{m}_0^f = \mathbf{m}_0$, $\mathbf{P}_0^f = \mathbf{P}_0$, transition matrices $\{\mathbf{A}_{i-1,i}, \mathbf{Q}_{i-1,i}\}_{i=1}^T$ and emission matrix $\mathbf{H}$.
Filtering:
for $i = 1, \ldots, T$ **do**
  Compute predictive filter distribution $p(\mathbf{s}_t|\tilde{\mathbf{Y}}_{1:i-1}) = N(\mathbf{s}_t|\mathbf{m}_t^p, \mathbf{P}_t^p)$:

$$\mathbf{m}_{t_i}^p = \mathbf{A}_{i-1,i}\mathbf{m}_{t_{i-1}}^f$$
$$\mathbf{P}_{t_i}^p = \mathbf{A}_{i-1,i}\mathbf{P}_{t_{i-1}}^f\mathbf{A}_{i-1,i}^\intercal + \mathbf{Q}_{i-1,i}.$$

  Let $\Lambda_{t_i} = \mathbf{H}\mathbf{P}_{t_i}^p\mathbf{H}^\intercal + \tilde{\mathbf{V}}_{t_i}$. Compute log marginal likelihood:

$$\ell_{t_i} = \log \mathbb{E}_{p(\mathbf{s}_{t_i}|\tilde{\mathbf{Y}}_{1:i-1})} N(\tilde{\mathbf{Y}}_{t_i}; \mathbf{H}\mathbf{s}_{t_i}, \tilde{\mathbf{V}}_{t_i}) = \log N(\tilde{\mathbf{Y}}_{t_i}; \mathbf{H}\mathbf{m}_{t_i}^p, \Lambda_{t_i})$$

  Compute updated filter distribution $p(\mathbf{s}_{t_i}|\tilde{\mathbf{Y}}_{1:i}) = N(\mathbf{s}_{t_i}|\mathbf{m}_{t_i}^f, \mathbf{P}_{t_i}^f)$:

$$\mathbf{W}_{t_i} = \mathbf{P}_{t_i}^p\mathbf{H}^\intercal\Lambda_{t_i}^{-1} \Rightarrow \mathbf{m}_{t_i}^f = \mathbf{m}_{t_i}^p + \mathbf{W}_{t_i}(\tilde{\mathbf{Y}}_{t_i} - \mathbf{H}\mathbf{m}_{t_i}^p), \quad \mathbf{P}_{t_i}^f = \mathbf{P}_{t_i}^p - \mathbf{W}_{t_i}\Lambda_{t_i}\mathbf{W}_{t_i}^\intercal.$$

end for
Smoothing, with initial conditions $\mathbf{m}_T^s = \mathbf{m}_T^f$, $\mathbf{P}_T^s = \mathbf{m}_T^f$:
for $i = T-1, \ldots, 1$ **do**
  Compute the smoothing distribution $q(\mathbf{s}_{t_i}) = p(\mathbf{s}_{t_i}|\tilde{\mathbf{Y}}_{1:T}) = N(\mathbf{s}_{t_i}|\mathbf{m}_{t_i}^s, \mathbf{P}_{t_i}^s)$ with the RTS smoother:

$$\mathbf{G}_{t_i} = \mathbf{P}_{t_i}^f\mathbf{A}_{i,i+1}[\mathbf{P}_{t_{i+1}}^p]^{-1} \Rightarrow \mathbf{m}_{t_i}^s = \mathbf{m}_{t_i}^f + \mathbf{G}_{t_i}(\mathbf{m}_{t_{i+1}}^s - \mathbf{m}_{t_{i+1}}^p), \quad \mathbf{P}_{t_i}^s = \mathbf{P}_{t_i}^f + \mathbf{G}_{t_i}(\mathbf{P}_{t_{i+1}}^s - \mathbf{P}_{t_{i+1}}^p)\mathbf{G}_{t_i}^\intercal$$

end for
Return: log marginal likelihood $\sum_{t=1}^T \ell_{t_i}$, marginal posteriors $\{q(\mathbf{s}_{t_i})\}_{i=1}^T$.

where $q(\mathbf{s}(\mathbf{R}, t_{1:T})) = N(\mathbf{s}(\mathbf{R}, t_{1:T})|m(\mathbf{R}, t_{1:T}), c(\mathbf{R}, t_{1:T}))$ is the approximate state posterior $s(\mathbf{R}, t_*)$ conditioned on $\mathbf{s}(\mathbf{R}, t_{1:T})$. Given in Tebbutt et al. (2021) and Wilkinson et al. (2021),

$$p(\mathbf{Z}(r_*, t_*)|\mathbf{s}(\mathbf{R}, t_*)) = N(\mathbf{Z}(r_*, t_*)|\mathbf{B}_{r_*}\mathbf{s}(\mathbf{R}, t_*), \mathbf{C}_{r_*}),$$

where

$$\begin{aligned}\mathbf{B}_{r_*} &= [\mathbf{K}^r_{r_*\mathbf{R}}(\mathbf{K}^r_{\mathbf{RR}})^{-1}] \otimes [\mathbf{L}^r_{\mathbf{RR}} \otimes \mathbf{H}^t] \\ \mathbf{C}_{r_*} &= k_t(0,0)[\mathbf{K}^r_{r_*r_*} - \mathbf{K}^r_{r_*\mathbf{R}}(\mathbf{K}^r_{\mathbf{RR}})^{-1}\mathbf{K}^r_{\mathbf{R}r_*}].\end{aligned}$$

Therefore the integral in Equation (10) yields $q(\mathbf{Z}(r_*, t_*)) = N(\mathbf{Z}(r_*, t_*)|\mathbf{B}_{r_*}m(t_{1:T}, \mathbb{R}), \mathbf{B}_{r_*}c(t_{1:T,\mathbb{R}})\mathbf{B}^\intercal_{r_*} + \mathbf{C}_{r_*})$.

B. Additional Experimental and Implementation Details

For the generating modelling tasks of corrupt image (re)-generation, the NLL on an test sequence $\mathbf{Y} = (\mathbf{Y}_1, \ldots, \mathbf{Y}_{100})$ is

$$\begin{aligned}\log p(\mathbf{Y}) &= \log \int p(\mathbf{Y}|\mathbf{Z})p(\mathbf{Z})\mathrm{d}\mathbf{Z} \\ &= \log \int p(\mathbf{Y}|\mathbf{Z}) \tfrac{p(\mathbf{Z})}{q(\mathbf{Z}|\mathbf{Y}_{\text{corrupt}})} q(\mathbf{Z}|\mathbf{Y}_{\text{corrupt}})\mathrm{d}\mathbf{Z} \\ &= \log \tfrac{1}{K}\textstyle\sum_{k=1}^K p(\mathbf{Y}|\mathbf{Z}_k) \tfrac{p(\mathbf{Z}_k)}{q(\mathbf{Z}_k|\mathbf{Y}_{\text{corrupt}})}, \quad \mathbf{Z}_k \sim q(\mathbf{Z}|\mathbf{Y}_{\text{corrupt}}), \quad k = 1 \ldots, K \\ &= \log \tfrac{1}{K} + \text{logsumexp}_{k=1,\ldots,K}\left[\log p(\mathbf{Y}|\mathbf{Z}_k) - \log \tfrac{q(\mathbf{Z}_k|\mathbf{Y})}{p(\mathbf{Z}_k)}\right]. \end{aligned} \tag{11}$$

For the tasks of missing frame imputation, given test sequence $\mathbf{Y} = (\mathbf{Y}_1, \ldots, \mathbf{Y}_{100})$, write $\mathbf{Y} \equiv (\mathbf{Y}, \mathbf{Y}^c)$, where $\mathbf{Y}^c$ are the unseen frames and $\mathbf{Y}^c$ are seen frames.

$$\log p(\mathbf{Y}) \equiv \log p(\mathbf{Y}^c|\mathbf{Y}) + \log p(\mathbf{Y}).$$

$\log p(\mathbf{Y})$ can be computed by Equation 11 and

$$\begin{aligned}\log p(\mathbf{Y}^c|\mathbf{Y}) &= \log \int p(\mathbf{Y}^c|\mathbf{Z}^c)p(\mathbf{Z}^c|\mathbf{Y})\mathrm{d}\mathbf{Z}^c \\ &= \log \int p(\mathbf{Y}^c|\mathbf{Z}^c)\left[\int p(\mathbf{Z}^c|\mathbf{Z})p(\mathbf{Z}|\mathbf{Y})\mathrm{d}\mathbf{Z}\right]\mathrm{d}\mathbf{Z}^c \\ &= \log \int p(\mathbf{Y}^c|\mathbf{Z}^c)\Bigg[\underbrace{\int p(\mathbf{Z}^c|\mathbf{Z})\underbrace{q(\mathbf{Z}|\mathbf{Y})}_{\text{encoder}}\mathrm{d}\mathbf{Z}}_{=:q(\mathbf{Z}^c|\mathbf{Y}) \approx p(\mathbf{Z}^c|\mathbf{Y})}\Bigg]\mathrm{d}\mathbf{Z}^c \\ &\approx \log \int p(\mathbf{Y}^c|\mathbf{Z}^c)q(\mathbf{Z}^c|\mathbf{Y})\mathrm{d}\mathbf{Z}^c \\ &= \log \tfrac{1}{K}\textstyle\sum_{k=1}^K p(\mathbf{Y}^c|\mathbf{Z}^c_k), \quad \mathbf{Z}^c_k \sim q(\mathbf{Z}^c|\mathbf{Y}), \quad k = 1 \ldots, K, \\ &= \log \tfrac{1}{K} + \text{logsumexp}_{k=1,\ldots,K} \log p(\mathbf{Y}^c|\mathbf{Z}^c_k),\end{aligned}$$

$q(\mathbf{Z}^c|\mathbf{Y})$ can be analytically obtained (e.g. GP posterior distribution). For VRNN/KVAE, $q(\mathbf{Z}^c|\mathbf{Y})$ would just be determined by the outputs of the RNN at each time step $t = 1, \ldots, 100$.

For the spatiotemporal modelling task, we compute the negative log predictive distribution (NLPD) at new spatial locations $s^* \notin \mathcal{S}$ and temporal locations $t \in \mathcal{T}$ observed during training, conditioned on observed data $\mathcal{D} = \{\mathbf{Y}(s,t)\}_{s\in\mathcal{S}, t\in\mathcal{T}}$, which is

$$\begin{aligned}\log p(\mathbf{Y}(t, s_*)|\mathcal{D}) &= \log \int p(\mathbf{Y}(t, s_*)|\mathbf{Z}(t, s_*))p(\mathbf{Z}(t, s_*)|\mathcal{D})\mathrm{d}\mathbf{Z}(t, s_*) \\ &\approx \log \int p(\mathbf{Y}(t, s_*)|\mathbf{Z}(t, s_*))q(\mathbf{Z}(t, s_*)|\mathcal{D})\mathrm{d}\mathbf{Z}(t, s_*) \\ &= \log \tfrac{1}{K}\textstyle\sum_{k=1}^K p(\mathbf{Y}(t, s_*)|\mathbf{Z}_k(t, s_*)), \quad \mathbf{Z}_k(t, s_*) \sim q(\mathbf{Z}(t, s_*)|\mathcal{D}), \quad k = 1, \ldots, K, \\ &= \log \tfrac{1}{K} + \text{logsumexp}_{k=1,\ldots,K} \log p(\mathbf{Y}(t, s_*)|\mathbf{Z}_k(t, s_*)).\end{aligned}$$

We hereby provide a derivation of Equation 11 for each model (using the original notation as much as possible).

MGPVAE:

$$\begin{aligned}
\log p(\mathbf{Y}) &= \log \textstyle\int p(\mathbf{Y}|\mathbf{s}) \frac{p(\mathbf{s})}{q(\mathbf{s})} q(\mathbf{s}) \mathrm{d}\mathbf{s}, \\
&= \log \textstyle\int p(\mathbf{Y}|\mathbf{s}) \frac{p(\mathbf{s}) \int p(\tilde{\mathbf{Y}}|\mathbf{H}\mathbf{s}, \tilde{\mathbf{V}}) p(\mathbf{s}) \mathrm{d}\mathbf{s}}{p(\tilde{\mathbf{Y}}|\mathbf{H}\mathbf{s}, \tilde{\mathbf{V}}) p(\mathbf{s})} q(\mathbf{s}) \mathrm{d}\mathbf{s}, \\
&\approx \log \tfrac{1}{K} + \log \textstyle\sum_{k=1}^{K} p(\mathbf{Y}|\mathbf{s}) \frac{\int p(\tilde{\mathbf{Y}}|\mathbf{H}\mathbf{s}_k, \tilde{\mathbf{V}}) p(\mathbf{s}_k) \mathrm{d}\mathbf{s}}{p(\tilde{\mathbf{Y}}|\mathbf{H}\mathbf{s}_k, \tilde{\mathbf{V}})}, \quad \mathbf{s}_k \sim q(\mathbf{s}_k) \\
&= \log \tfrac{1}{K} + \text{logsumexp}_{k=1,\ldots,K} \log p(\mathbf{Y}|\mathbf{s}_k) - \log p(\tilde{\mathbf{Y}}|\mathbf{H}\mathbf{s}_k, \tilde{\mathbf{V}}) + \log \mathbb{E}_{p(\mathbf{s})} N(\tilde{\mathbf{Y}}|\mathbf{H}\mathbf{s}, \tilde{\mathbf{V}})
\end{aligned}$$

KVAE: When encountering a missing frame, the Kalman gain is zero in the Kalman filtering and smoothing operations. The filter prediction distribution is then used to recompute the A and C matrices, which are finally fed back into the usual filtering algorithm.

$$\begin{aligned}
\log p(\mathbf{Y}) &= \log \textstyle\int \frac{p(\mathbf{Y}|\mathbf{a}) p(\mathbf{a}|\mathbf{Z}) p(\mathbf{Z}) q(\mathbf{a}, \mathbf{Z}|\mathbf{Y})}{q(\mathbf{a}, \mathbf{Z}|\mathbf{Y})} \mathrm{d}\mathbf{a}\mathrm{d}\mathbf{Z}, \quad \left[q(\mathbf{a}, \mathbf{Z}|\mathbf{Y}) = q(\mathbf{a}|\mathbf{Y}) p(\mathbf{Z}|\mathbf{a}) \right] \\
&\approx \log \textstyle\sum_{k=1,\ldots K} \frac{p(\mathbf{Y}|\mathbf{a}_k) p(\mathbf{a}_k|\mathbf{Z}_k) p(\mathbf{Z}_k)}{q(\mathbf{a}_k|\mathbf{Y}_k) p(\mathbf{Z}_k|\mathbf{Y}_k)} + \log \tfrac{1}{K}, \quad (\mathbf{a}_k, \mathbf{Z}_k) \sim q(\mathbf{a}_k, \mathbf{Z}_k|\mathbf{Y}) \\
&= \log \tfrac{1}{K} + \text{logsumexp}_{k=1,\ldots,K} \log p(\mathbf{Y}|\mathbf{a}_k) - \log q(\mathbf{a}_k|\mathbf{Y}) + \log p(\mathbf{a}_k|\mathbf{Z}_k) + \log p(\mathbf{Z}_k) - \log p(\mathbf{Z}_k|\mathbf{a}_k).
\end{aligned}$$

VRNN: When encountering a missing frame, the previous hidden state is fed into prior network and $\mathbf{Z}_t$ is sampled from the prior. It is then decoded and the decoded output is then used as a pseudo-datapoint for autoencoding.

$$\begin{aligned}
\log p(\mathbf{Y}) &= \log \textstyle\prod_{t=1}^{T} p(\mathbf{Y}_t|\mathbf{Y}_{<t}) \\
&= \log \textstyle\int \prod_{t=1}^{T} p(\mathbf{Y}_t|\mathbf{Y}_{<t}, \mathbf{Z}_{\leq t}) \frac{p(\mathbf{Z}_t|\mathbf{Y}_{<t}, \mathbf{Z}_{<t})}{q(\mathbf{Z}_t|\mathbf{Y}_{\leq t}, \mathbf{Z}_{<t})} q(\mathbf{Z}_t|\mathbf{Y}_{\leq t}, \mathbf{Z}_{<t}) \mathrm{d}\mathbf{Z}, \\
&\approx \log \tfrac{1}{K} + \log \textstyle\sum_{k=1}^{K} \prod_{t=1}^{T} p(\mathbf{Y}_t|\mathbf{Y}_{<t}, \mathbf{Z}_{\leq t}^k) \frac{p(\mathbf{Z}_t^k|\mathbf{Y}_{<t}\mathbf{Z}_{<t}^k)}{q(\mathbf{Z}_t^k|\mathbf{Y}_{\leq t}, \mathbf{Z}_{<t}^k)}, \quad \mathbf{Z}_{1:T}^k \sim q(\mathbf{Z}_{\leq T}|\mathbf{Y}_{\leq T}) = \textstyle\prod_{t=1}^{T} q(\mathbf{Z}_t|\mathbf{Y}_{\leq t}, \mathbf{Z}_{<t}) \\
&= \log \tfrac{1}{K} + \text{logsumexp}_{k=1,\ldots,K} \textstyle\sum_{t=1}^{T} \log p(\mathbf{Y}_t|\mathbf{Y}_{<t}, \mathbf{Z}_{\leq t}^k) - \log \frac{q(\mathbf{Z}_t^k|\mathbf{Y}_{\leq t}\mathbf{Z}_{<t}^k)}{p(\mathbf{Z}_t^k|\mathbf{Y}_{<t}, \mathbf{Z}_{<t}^k)}
\end{aligned}$$

LatentODE: The latentODE places a prior over the initial conditions $\mathbf{Z}_0 \sim p(\mathbf{Z}_0)$ and a posterior via ODERNN $q(\mathbf{Z}_0|\mathbf{Y}_{1:T}) \equiv q(\mathbf{Z}_0)$. Thus the NLL is

$$\begin{aligned}
\log p(\mathbf{Y}) &= \log \textstyle\int p(\mathbf{Y}|\mathbf{Z}_0) \frac{p(\mathbf{Z}_0)}{q(\mathbf{Z}_0)} q(\mathbf{Z}_0) d\mathbf{Z} \\
&\approx \log \tfrac{1}{K} + \text{logsumexp}_{k=1,\ldots,K} \log p(\mathbf{Y}|\mathbf{Z}_0^k) - \log \tfrac{q(\mathbf{Z}_0^k)}{p(\mathbf{Z}_0^k)}, \quad \mathbf{Z}_0^k \sim p(\mathbf{Z}_0).
\end{aligned}$$

GPVAE:

$$\begin{aligned}
\log p(\mathbf{Y}) &= \log \textstyle\int p(\mathbf{Y}|\mathbf{Z}) \frac{p(\mathbf{Z})}{q(\mathbf{Z})} q(\mathbf{Z}) d\mathbf{Z} \\
&\approx \log \tfrac{1}{K} + \text{logsumexp}_{k=1,\ldots,K} \log p(\mathbf{Y}|\mathbf{Z}_k) - \log \tfrac{q(\mathbf{Z}_k|\mathbf{Y})}{p(\mathbf{Z}_k)}, \quad \mathbf{Z}_k \sim q(\mathbf{Z}_k|\mathbf{Y}).
\end{aligned}$$

SVGPVAE: We are given that $q(\mathbf{Z}, \mathbf{G}_m) = p(\mathbf{Z}|\mathbf{G}_m) q(\mathbf{F}_m|\mu, \mathbf{A})$, where $\mathbf{G}_m$ is the inducing variable and $(\mu, \mathbf{A})$ are the inducing variable mean and covariance.

$$\begin{aligned}
\log p(\mathbf{Y}) &= \log \textstyle\int p(\mathbf{Z}, \mathbf{G}_m, \mathbf{Y}) \frac{q(\mathbf{Z}, \mathbf{G}_m)}{q(\mathbf{Z}, \mathbf{G}_m)} \mathrm{d}\mathbf{Z}\mathrm{d}\mathbf{G}_m \\
&= \log \textstyle\int \frac{p(\mathbf{Y}|\mathbf{Z}) p(\mathbf{Z}|\mathbf{G}_m) p(\mathbf{G}_m)}{p(\mathbf{Z}|\mathbf{G}_m) q(\mathbf{G}_m)} p(\mathbf{Z}|\mathbf{G}_m) q(\mathbf{G}_m) \mathrm{d}\mathbf{Z}\mathrm{d}\mathbf{G}_m \\
&= \log \textstyle\int \frac{p(\mathbf{Y}|\mathbf{Z}) p(\mathbf{G}_m)}{q(\mathbf{G}_m)} p(\mathbf{Z}|\mathbf{G}_m) q(\mathbf{G}_m) \mathrm{d}\mathbf{Z}\mathrm{d}\mathbf{G}_m \\
&\approx \tfrac{1}{P} \textstyle\sum_{p=1}^{P} \log \tfrac{1}{K} + \text{logsumexp}_{k=1,\ldots,K} \log p(\mathbf{Y}|\mathbf{Z}_k^p) - \log \frac{q(\mathbf{G}_m^k)}{p(\mathbf{G}_m^k)}, \quad \mathbf{G}_m^k \sim q(\mathbf{G}_m) \quad \mathbf{Z}_k^p \sim p(\mathbf{Z}_k|\mathbf{G}_m^k),
\end{aligned}$$

where $p(\mathbf{Z}_k|\mathbf{G}_m)$ is a standard GP conditional distribution:

$$p(\mathbf{Z}_k|\mathbf{G}_m) \sim N\Big(\mathbf{Z}_k|\mathbf{K}_{xm}\mathbf{K}_{mm}^{-1}\mathbf{G}_m, \mathbf{K}_{xx} - \mathbf{K}_{xm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mx}\Big),$$
$$q(\mathbf{G}_m) \equiv q(\mathbf{G}_m|\mu, \mathbf{A}) \sim N\Big(\mathbf{G}_m|\mu, \mathbf{A}\Big).$$

B.1. Video Data

For each of the 4000 train and 1000 test MNIST images, we create a $T = 100$ length sequence of rotating MNIST images by applying a clockwise rotation with period $t = 50$, meaning that the image gets rotated twice. For corrupt video sequences, we randomly remove 60% of the pixels and replace them with the value 0. For the missing frames imputation task, we randomly remove 60% of the frames and keep track of the time steps for which the frames are removed.

For each model during training, we used the following configurations (note that we use the standard PyTorch-TensorFlow-JAX notation for network layers).

- Batch size: 40
- Training epochs: 300
- Number of latent channels: $L = 16$
- Adam optimizer learning rate: 1e-3
- clipGradNorm(model parameters, 100)
- Encoder structure: Conv(out=32, k=3, strides=2), ReLU(), Conv2D(out=32, k=3, strides=2), ReLU(), Flatten(), hiddenToVariationalParams(), where hiddenToVariationalParams() depends on the model.
- Decoder structure: Linear(L, 8*8*32), Reshape((8,32,32)), Conv2DTranspose(out=64, k=3, strides=2, padding=same), ReLU(), Conv2DTranspose(out=32, k=3, strides=2, padding=same), ReLU(), Conv2DTranspose(out=1, k=3, strides=1, padding=same), Reshape((32, 32, 1)).
- If KVAE or VRNN, we had to applying KL and Adam optimizer step size schedulers in order to make them work, meaning that they are more intricately optimised than the other models.
- If GPVAE and MGPVAE, initialise the kernel lengthscales with 40 for each latent channel. The kernel hyperparameters (lengthscales and scales) are subsequently optimised via the ELBO.
- The model at the last training epoch is used for test evaluation.

Remark: We hereby explain why GPVAE (Fortuin et al., 2020) is not practically suited for the missing frames imputation task. Recall that their approximate posterior is defined as $q(z_{1:T}|x_{1:T}) = N(m_j, \Lambda_j^{-1})$, where $\Lambda_j = B_j^\intercal B_j$ and B_j is an upper triangular band matrix, parameterised by outputs from the encoder. For a batch of sequences with varying lengths after removing missing frames, B_j and hence Λ_j will be padded with zeros (assuming we rearrange the orders of the time points). This is because most mainstream deep learning libraries (e.g. JAX, PyTorch and TensorFlow) all require static shape arrays/tensors and batch operations can only be done when all the arrays/tensors are of the same shape. However, we are not aware of any computationally efficient methods to "invert" a batch of Λ_j's with varying 0-padding sizes. Whilst it is still feasible to implement GPVAE for missing data imputation tasks, either the batch size will have to be 1 or we will need an extra for loop to iterate through each of the sequences in each batch during training. This severely constrains the practical computational efficiency of GPVAE, rendering it unsuitable as a missing frames imputation model.

Implementation Details: VRNN and KVAE are implemented in PyTorch. GPVAE mostly a non-modified version as the original TensorFlow implementation in Fortuin et al. (2020). MGPVAE is implemented in JAX using Objax (Developers, 2020). One issue in Objax is that the convolutional operator only supports float32 precision (otherwise the implementation will be very slow) and therefore we need to manually cast the CNN weights and inputs into float32 (from float64). This is a slight disadvantage of MGPVAE that hopefully can be resolved in the future through further advances in JAX-based CNN implementations.

During test time, we used $K = 20$ latent samples to compute the NLL and RMSE (via the posterior mean). The NLL and RMSE of the predictive results on the test set are calculated over the entire sequence.

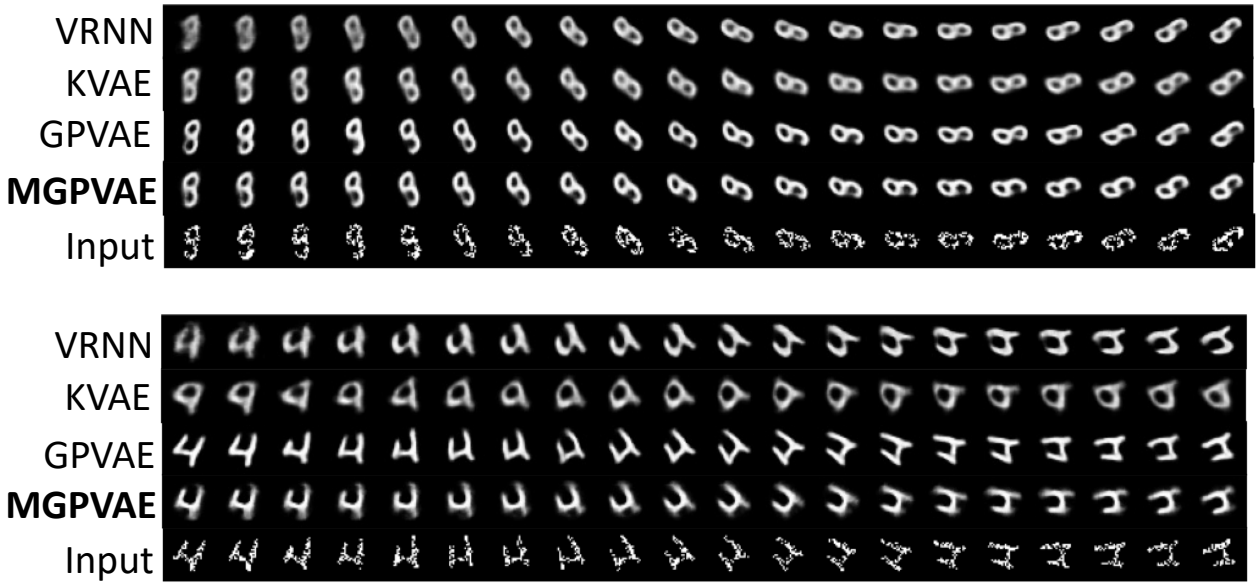

Figure 8: Additional corrupt frames imputation results.

B.2. Mujoco Action Data

We create missing time steps by randomly dropping out 60% of them.

For each model during training, we used the following configurations (note that we use the standard PyTorch-TensorFlow-Jax notation for network layers). Note that unlike Rubanova et al. (2019), we do not use a masked autoencoder approach for training and only use the non-missing time steps to compute the ELBO.

- Batch size: 16
- Training epochs: 1000 if $T = 100$ and 500 if $T = 1000$.
- Number of latent channels: $L = 15$
- Adam optimizer learning rate: 1e-3.
- clipGradNorm(model parameters, 100)
- Encoder structure: Linear(L, 32)-ReLU-Linear(32, L). If latentODE then slightly different but similar due to the ODE-RNN.
- Decoder structure: Linear(L, 16)-ReLU-Linear(16, 14).
- If KVAE or VRNN, we had to applying KL and Adam optimizer step size schedulers in order to make them work, meaning that they are more intricately optimised than the other models. LatentODE also used a KL scheduler.
- If SVGPVAE or MGPVAE, initialise the kernel lengthscales with: 5 if $T = 100$ and 50 if $T = 1000$. The kernel hyperparameters (lengthscales and scales) are subsequently optimised via the ELBO.
- SVGPVAE, VRNN, latentODE and MGPVAE we use early stopping with the validation RMSE on 320 if $T = 100$ else 80 validation sequences (not part of train or test sets). For KVAE and CRU, adding cross-validation is practically too time consuming and so the model at the last training epoch is used for test evaluation.

Remark: CRU (Schirmer et al., 2022) is capable of modelling continuous data in the form $(t_i, y_i)_{i=0}^N$, but, unlike latentODE, SVGPVAE or MGPVAE, it is unable to condition on $(t_i, y_i)_i$ and then predict at any time step $t \in (t_0, t_N)$. Therefore to train the CRU for our task, we have to treat it the same way as VRNN and KVAE, where we perform Kalman filtering

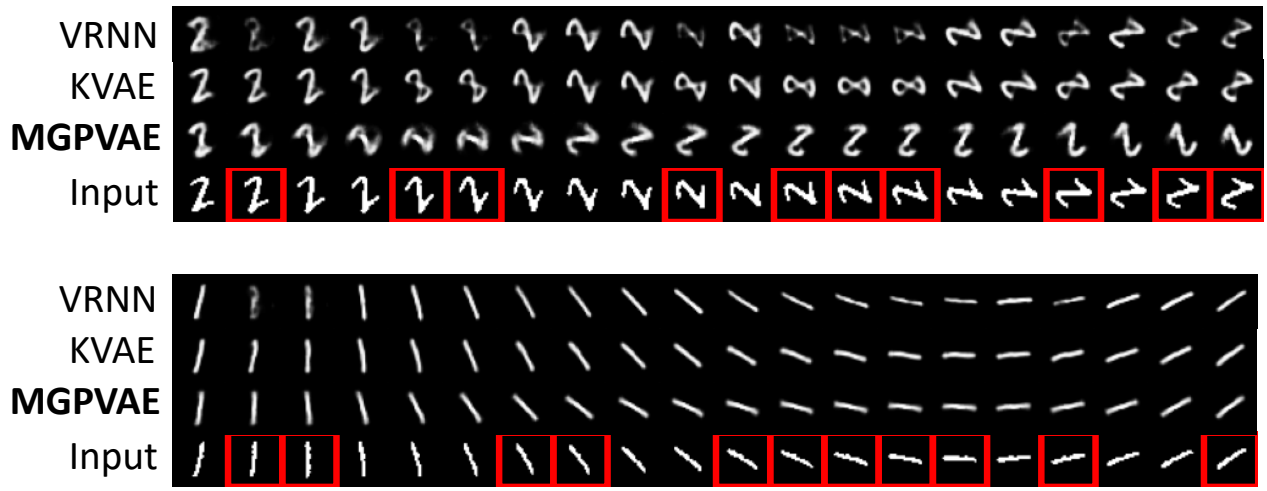

Figure 9: Additional missing frames imputation results.

in regular time steps and then mask out the unobserved time steps during training. Furthermore, this model is trained via maximum likelihood estimation, and so we only report the test RMSE metric.

Implementation Details: VRNN, KVAE, latentODE, SVGPVAE and CRU are implemented in PyTorch. For latentODE and CRU, they are mostly unmodified, based off the original author's implementations. We rewrite SVGPVAE using the more efficient framework of Functorch (Horace He, 2021), which we find to give better implementation efficiency and performance than the original TensorFlow implementation. MGPVAE is implemented with Objax Developers (2020) within JAX.

During test time, we use $K = 20$ latent samples to compute the NLL and RMSE (via the posterior mean). The NLL and RMSE of the predictive results on the test set are calculated over the entire sequence.

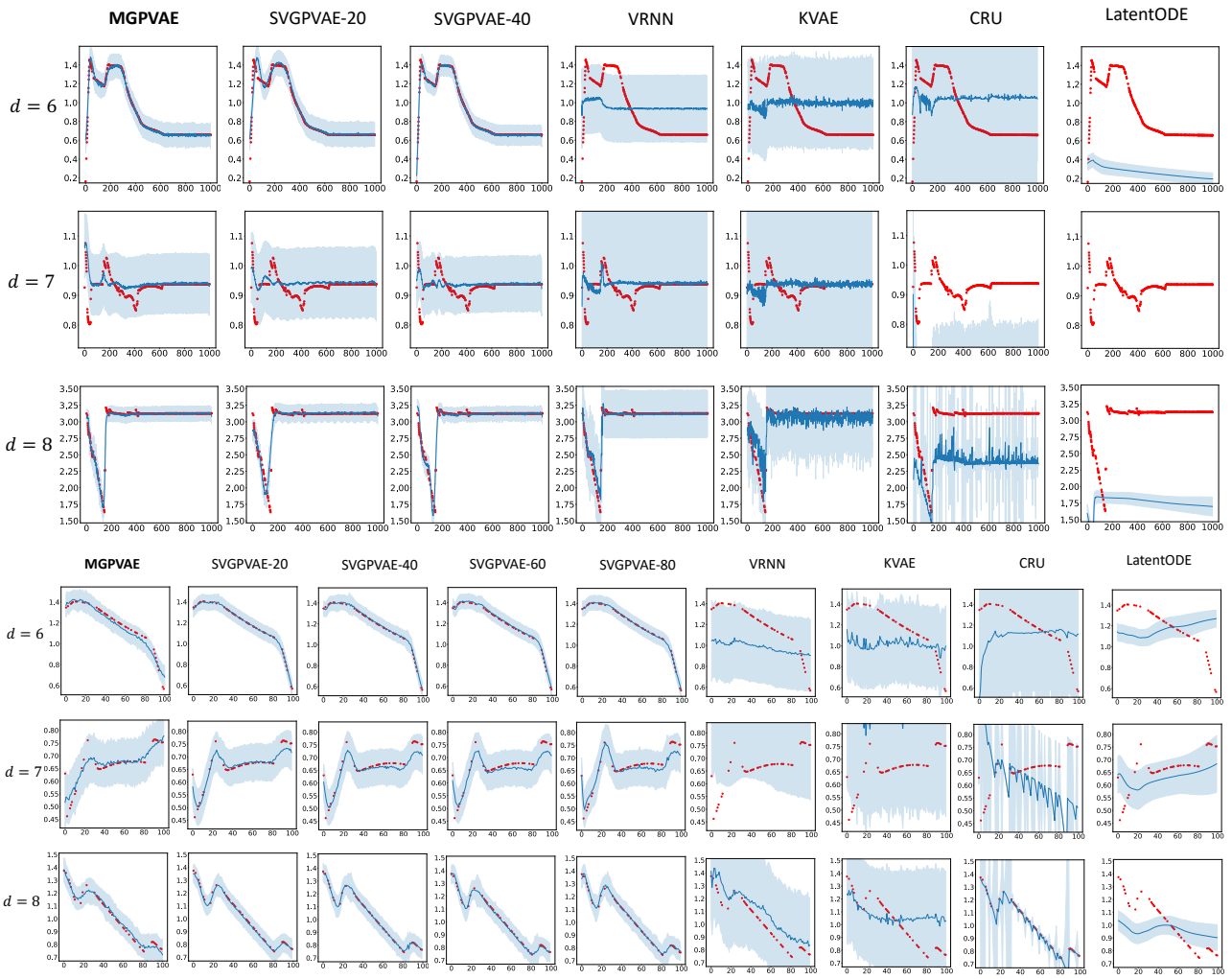

Figure 10: Additional results figures for unseen Mujoco sequences. Note that some predictions are not showing as they fall outside the graph limits.

B.3. Spatiotemporal Data

We download the ECMWF ERA5 data (ECMWF/ERA5/DAILY) from Google Earth Engine API (Gorelick et al., 2017). The data vectors contain:

- Mean 2m air temperature
- Minimum 2m air temperature
- Maximum 2m air temperature
- Dewpoint 2m air temperature
- Surface pressure
- Mean sea level pressure
- u component of wind 10m
- v component of wind 10m

The spatial locations are represented by [longitude, latitude, elevation]. We index time by $0, 1, \ldots, 100$.

The model configurations for SVGPVAE and MGPVAE are:

- clipGradNorm(model parameters, 100)
- Encoder structure: Linear(L, 16)-ReLU-Linear(16, L).
- Decoder structure: Linear(L, 16)-ReLU-Linear(16, 14).
- The model at the last training epoch is used for test evaluation.

Implementation Details: MOGP is implemented using GPFlow (Matthews et al., 2017) within TensorFlow. SVGPVAE is again implemented in PyTorch with Functorch. MGPVAE is implemented with Objax within JAX.

MOGP and SVGPVAE: We stack time and space together in the dataset, and have inducing points over space-time jointly. During training, we use only 1 latent sample to compute the ELBO for SVGPVAE, a minibatch size of 40 and 200 epochs over the entire dataset (this gives roughly 20,000 gradient steps). We use an Adam optimizer learning rate of 1e-3. Similary for MOGP, we train over 20000 epochs of minibatches of size 40.

MGPVAE: We don't do any minibatching and directly use the entire training dataset, which is of dimension roughly 60-100-8. During training, we compute the ELBO with 20 latent samples and train over 20000 epochs. We use an Adam optimizer learning rate of 1e-3.

Prediction: We compute the NLL with 20 latent samples over a test set of unseen spatial locations at all time steps.

C. Possible Future Developments

It is possible to extend MGPVAE to incorporate temporal and spatial inducing points, by the virtue of the works of Adam et al. (2020) and Hamelijnck et al. (2021). The main challenge would be forming the likelihood approximation, which would have to rely on set encoders such as DeepSets (Zaheer et al., 2017) or Set Transformer (Lee et al., 2019) to output approximate likelihoods factorised over inducing points (instead of over time points). For temporal data, the main challenge would be to cluster the time points together for each inducing time location. For spatiotemporal data, the clustering problem extends to space-time clusters.

The state space may also be enriched via normalising flows (Lin & Yin, 2023; Maroñas & Hernández-Lobato, 2022), which may additionally enrich the latent dynamics or simplify the learning dynamics. Another approach to enforcing global properties without GPs would be via a GAM-type model, such as that of Prophet (Taylor & Letham, 2018).

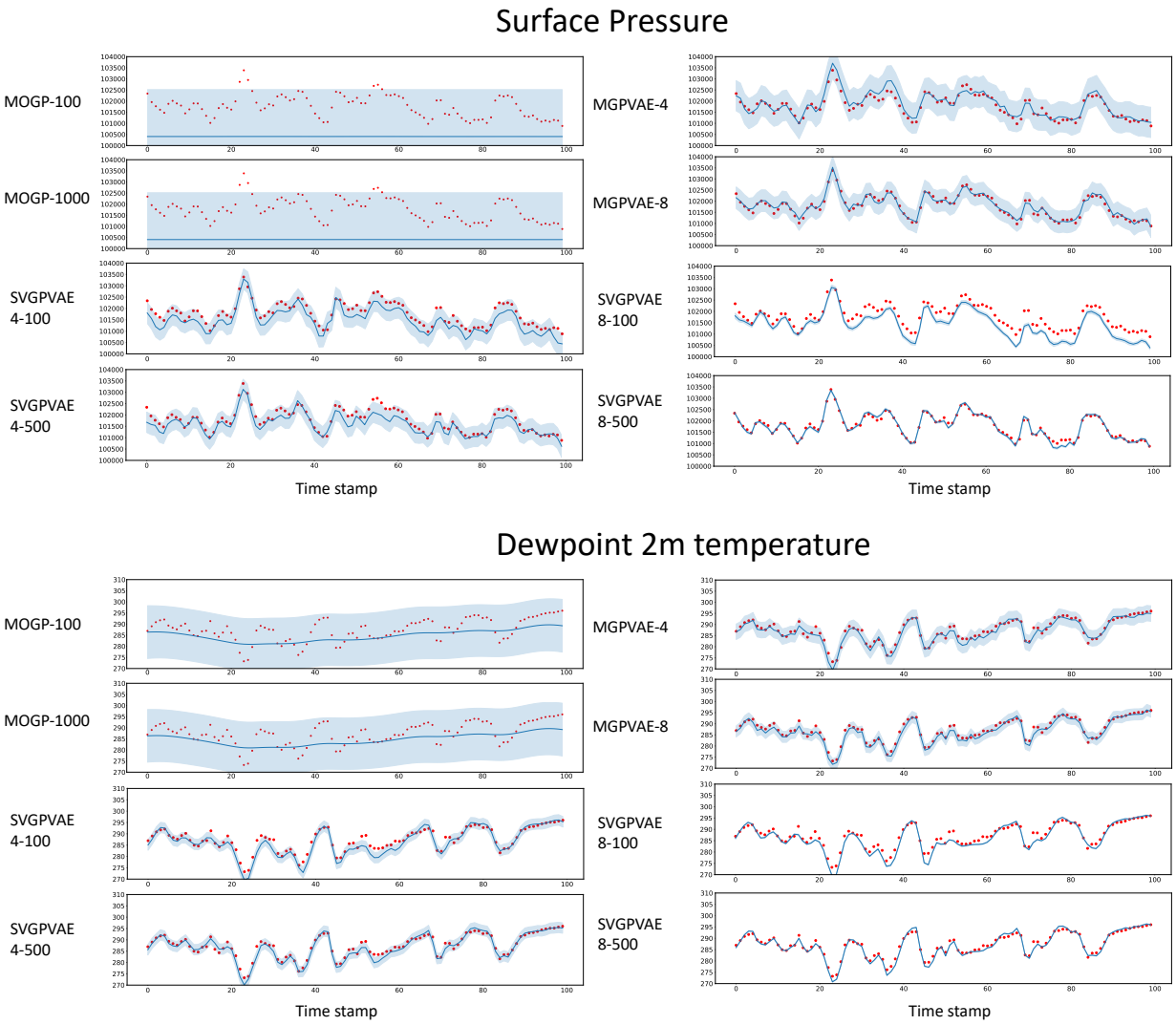

Figure 11: Additional figures of the results for the ERA5 experiment given fixed spatial locations.