# Inter-domain Inference for Gaussian Process Variational Autoencoders

**Xinxing Shi** **Xiaoyu Jiang** **Mauricio A. Álvarez**
Department of Computer Science
The University of Manchester
Manchester, United Kingdom
{xinxing.shi, xiaoyu.jiang, mauricio.alvarezlopez}@manchester.ac.uk

## Abstract

Gaussian Process Variational Autoencoders (GPVAEs) effectively model sample dependencies via latent GP priors, but their inference remains computationally prohibitive at scale. Existing methods typically rely on inducing points, which are restricted to evaluations in the input domain and provide limited control over inductive bias. We introduce a scalable inference framework for GPVAEs that uses inducing variables defined as general linear functionals of the latent GPs, rather than point evaluations. Our formulation recovers the standard inducing-point GPVAE inference as a special case and yields a simplified reconstruction-minus-KL training objective that makes this connection explicit. More broadly, it provides a principled interface for incorporating operator-dependent representations and computational structures via the choice of inducing features. In particular, we instantiate this framework using Fourier features to capture global spectral correlations and B-spline features to induce efficient banded covariances. Across multiple tasks in representation learning, imputation, and conditional generation, our method offers a competitive accuracy–efficiency trade-off to existing approaches.

## 1 Introduction

Variational Autoencoders (VAEs; Kingma and Welling, 2014) typically employ independent Gaussian priors over sample-specific latent variables, which may miss dependencies across observations. Gaussian Process Variational Autoencoders (GPVAEs; Casale et al., 2018) address this limitation by placing Gaussian Process (GP) priors over latent variables indexed by low-dimensional auxiliary inputs, thereby modelling correlations across data points. This makes GPVAEs well-suited for applications where informative auxiliary inputs are available and explicit latent dependence modelling is desired, such as video prediction (Bhagat et al., 2020), protein reconstruction (Moreta et al., 2022), and neural-data modelling (Gondur et al., 2024).

Despite their practical success, GPVAEs remain computationally challenging because exact inference in the latent GPs scales cubically with the number of data points (Rasmussen and Williams, 2006). A widely used way to mitigate this bottleneck is to use inducing-point methods (Titsias, 2009; Hensman et al., 2015), which approximate the latent GP posterior through a smaller set of inducing variables (Ashman et al., 2020; Jazbec et al., 2021). However, existing inducing-point GPVAEs typically define inducing variables as point evaluations, which require optimising inducing locations in the input domain and offer limited control over the inductive bias and representational structure of the approximation. These limitations motivate moving beyond inducing point-based constructions in GPVAE inference.

Inspired by the inter-domain GP perspective (Lázaro-Gredilla and Figueiras-Vidal, 2009; van der Wilk et al., 2020), we therefore present a scalable inference framework for GPVAE models, *Inter-*

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

*Domain GPVAE* (IDGPVAE). We focus on challenging modelling scenarios where factorised priors struggle, and explicit, structured latent modelling becomes beneficial. By defining inducing variables as general linear functionals of the latent GPs, our method naturally integrates inter-domain GP approximations with deep generative models. This extends conventional inducing-point formulations and provides a more flexible variational interface for GPVAE inference. Depending on the choice of linear functionals, the latent approximation can inherit tailored inductive biases and, when available, exploitable algebraic structures. Additionally, when these functionals are fixed a priori, no separate optimisation of inducing locations is required.

Our main contributions are as follows:

- We introduce an inter-domain framework for scalable GPVAE inference where the inducing variables are constructed via linear operators, enabling inducing features with tailored inductive bias and, with suitable operators, exploitable computational structures.
- Our framework yields a unified training objective and recovers the standard inducing-point GPVAE as a special case.
- We study two representative instantiations of the proposed framework: Fourier features and B-spline features. These choices demonstrate that our method enables operator-adapted representations, thereby providing more efficient latent modelling than inducing-point approaches.
- We provide practical implementations using custom autograd-compatible banded operators. Empirical evaluations on latent representation learning, data imputation, and conditional generation benchmarks show that our method achieves a competitive accuracy–efficiency trade-off to its counterparts.

## 2 Background

### 2.1 GPVAE setup and notation

A GP $z : \mathbb{R}^D \to \mathbb{R}$ is a stochastic process such that any finite collection of function values at inputs $\mathbf{X} = [\mathbf{x}_n]_{n=1}^N$ is jointly Gaussian with covariance defined by a kernel $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$:

$$z(\cdot) \sim \mathcal{GP}(0, k(\cdot, \cdot')) \implies \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{zz}}), \quad \text{with} \quad \mathbf{z} \coloneqq [z(\mathbf{x}_n)]_{n=1}^N.$$

Here, the random vector $\mathbf{z} \in \mathbb{R}^N$ collects the function values and $\mathbf{K}_{\mathbf{zz}} = k(\mathbf{X}, \mathbf{X})$ is the covariance matrix characterising the correlation between any input pairs.

GPVAE integrates the GP prior into VAE. Consider $N$ paired data points where $\mathbf{y}_n \in \mathbb{R}^K$ denotes the $n^{\text{th}}$ observation and $\mathbf{x}_n \in \mathbb{R}^D$ is the corresponding (low-dimensional) auxiliary information. Let

$$\mathbf{Y} = [\mathbf{y}_n]_{n=1}^N \in \mathbb{R}^{N \times K}, \quad \mathbf{X} = [\mathbf{x}_n]_{n=1}^N \in \mathbb{R}^{N \times D}, \quad \mathbf{Z} = [\mathbf{z}_n]_{n=1}^N \in \mathbb{R}^{N \times L},$$

where $\mathbf{z}_n \in \mathbb{R}^L$ is the latent vector associated with the sample $n$. In GPVAE, each latent channel $\mathbf{z}^l \coloneqq z^l(\mathbf{X}) \in \mathbb{R}^N$ (i.e., the $l^{\text{th}}$ column of $\mathbf{Z}$[1]) follows a GP prior and is then fed into a decoder network $p_\theta$. The generative process is given by

$$p_\psi(\mathbf{Z} \mid \mathbf{X}) = \prod_{l=1}^L p_\psi(\mathbf{z}^l \mid \mathbf{X}) = \prod_{l=1}^L \mathcal{N}(\mathbf{z}^l \mid \mathbf{0}, \mathbf{K}_{\mathbf{zz}}^l), \quad p_\theta(\mathbf{Y} \mid \mathbf{Z}) = \prod_{n=1}^N p_\theta(\mathbf{y}_n \mid \mathbf{z}_n), \tag{1}$$

where $\psi$ and $\theta$ are the kernel and decoder parameters, respectively. Thus, *within* each latent channel, the GP prior introduces correlations across samples via the auxiliary inputs, while *cross-channel* dependence is captured through the decoder. Compared with VAE, the inference now involves dense $N \times N$ covariance matrices induced by the prior $p_\psi(\mathbf{Z} \mid \mathbf{X})$ in the Evidence Lower Bound (ELBO).

### 2.2 Inducing-point GPVAE inference

A convenient inference scheme for GPVAE is obtained through pseudo-observations produced by the encoder (Pearce, 2020; Zhu et al., 2023). Specifically, for each latent channel $l$ and sample $n$, the encoder $\tilde{q}_\varphi$ takes $\mathbf{y}_n$ as input and parametrises a Gaussian site factor

$$\tilde{q}_\varphi(z_n^l \mid \mathbf{y}_n) = \mathcal{N}(z_n^l \mid \mu_n^l(\mathbf{y}_n), \sigma_n^{2,l}(\mathbf{y}_n)), \tag{2}$$

[1]We adopt a consistent $z$-based notation throughout the main text to maintain the connection among the latent GP function $z^l(\cdot)$, its finite evaluations $\mathbf{z}^l = z^l(\mathbf{X})$, and the datum-wise latent vector $\mathbf{z}_n$.

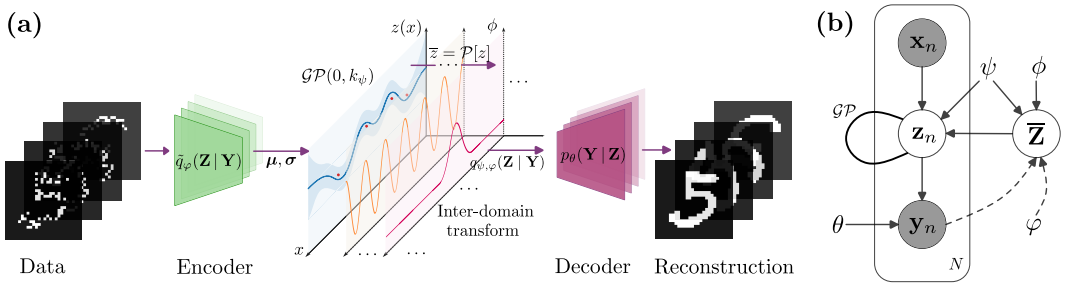



**Figure 1: (a)** IDGPVAE constructs inducing variables through linear operators $\mathcal{P}$ applied to the latent GP $z(\cdot)$ (e.g., projections onto basis functions). **(b)** Graphical model of IDGPVAE. Solid/dashed lines denote the generative/encoding parts, respectively. The thick self-loop denotes the latent GP.

where $z_n^l$ is the $n^{\text{th}}$ element of $\mathbf{z}^l$, and $\varphi$ denotes the encoder parameters. Combining the latent GP prior and the encoder distribution above yields a conjugate variational posterior for the latent variables

$$q_{\psi,\varphi}(\mathbf{Z} \mid \mathbf{Y}, \mathbf{X}) = \prod_{l=1}^{L} \frac{p_\psi(\mathbf{z}^l \mid \mathbf{X}) \prod_{n=1}^{N} \tilde{q}_\varphi(z_n^l \mid \mathbf{y}_n)}{\mathcal{Z}_{\psi,\varphi}^l(\mathbf{Y}, \mathbf{X})} \propto p_\psi(\mathbf{Z})\, \tilde{q}_\varphi(\mathbf{Z} \mid \mathbf{Y}), \tag{3}$$

where the encoder distribution $\tilde{q}_\varphi$ can be viewed as a heteroscedastic pseudo-likelihood. The normalising constant $\mathcal{Z}_{\psi,\varphi}^l$ has a closed-form Gaussian expression. As a result, the main computational burden shifts to evaluating $\mathcal{Z}_{\psi,\varphi}^l$, which has $\mathcal{O}(N^3)$ complexity.

To circumvent this cubic cost from latent GPs, current scalable inference methods combine the pseudo-observation view above with a sparse variational approximation based on inducing points (Titsias, 2009; Hensman et al., 2013), as exemplified by SVGPVAE (Jazbec et al., 2021). In this setting, each latent channel introduces $M$ ($\ll N$) inducing variables $\bar{\mathbf{z}}^l$, which are tied to point evaluations of the same latent GP at trainable inducing locations $\bar{\mathbf{X}} = [\bar{\mathbf{x}}_m]_{m=1}^M \in \mathbb{R}^{M \times D}$:

$$\bar{\mathbf{z}}^l \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}^l), \quad \text{with} \quad \bar{\mathbf{z}}^l \coloneqq [z^l(\bar{\mathbf{x}}_m)]_{m=1}^M,\ \mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}^l = k_\psi^l(\bar{\mathbf{X}}, \bar{\mathbf{X}}).$$

Since the inducing variables $\bar{\mathbf{z}}^l$ and the latent variables $\mathbf{z}^l$ are jointly Gaussian, they support a sparse GP approximation to the latent posterior. Jazbec et al. (2021) replace the normalising constant with its sparse variational bound, yielding a mini-batch training objective with *additional* terms and trainable $\bar{\mathbf{X}}$ in the same input domain (see Appendix A.2 for detailed derivation). At the same time, the inducing-point formulation above restricts inducing variables to point evaluations of the latent GPs. These considerations motivate our inter-domain inference framework for GPVAEs, which we introduce in the next section.

## 3 Inter-domain Gaussian process variational autoencoder

Our approach integrates inter-domain GP approximations into the latent space of deep generative models and broadens the inducing-variable design space of GPVAEs beyond point evaluations. In this section, we construct the GPVAE latent inducing variables as linear functionals of the latent GPs. Figure 1 illustrates the proposed model and its probabilistic graphical structure.

### 3.1 Inter-domain inducing variables for latent GPs

For notational clarity, we first describe the construction for a single latent channel. We suppress the channel superscript $l$ and explicit dependence on $\mathbf{X}$ in distributions when clear from context. Let the latent trajectory be governed by a GP prior $z(\cdot) \sim \mathcal{GP}(0,\, k_\psi)$, and $\mathbf{z} = z(\mathbf{X}) \in \mathbb{R}^N$ be its evaluations at the observed auxiliary coordinates. We introduce $M$ linear operators $\{\mathcal{P}_m\}_{m=1}^M$ and define the corresponding inducing variables by applying these operators to the original GP

$$\bar{z}_m \coloneqq \mathcal{P}_m[z], \qquad \bar{\mathbf{z}} \coloneqq [\bar{z}_m]_{m=1}^M \in \mathbb{R}^M.$$

We refer to the operator $\mathcal{P}_m$ as an *inducing feature*. One common example of inducing features is an integral operator associated with a feature function $\phi_m$, i.e., $\bar{z}_m = \int_{\mathbb{R}^D} z(\mathbf{x})\phi_m(\mathbf{x})\,\mathrm{d}\mathbf{x}$. Conventional

inducing points are retrieved when $\phi_m$ is chosen as a Dirac delta function. More broadly, the operators may represent point evaluations, derivatives, convolutions, or other linear measurements of the latent process. This allows scalable inference in an informative representation space better aligned with prior structural assumptions about the latent process.

Since GPs are closed under linear transformations, the joint distribution of $\{\mathbf{z}, \bar{\mathbf{z}}\}$ remains Gaussian, with the covariance blocks determined by the chosen operators. In particular,

$$[\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}]_{m,n} = \mathcal{P}_m[k(\cdot, \mathbf{x}_n)], \qquad [\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}]_{m,m'} = \mathcal{P}_m[k(\cdot,\cdot')]\,\mathcal{P}_{m'}.$$

Hence, conditional on $\bar{\mathbf{z}}$, the latent variables follow the Gaussian conditioning formula

$$p_\psi(\mathbf{z} \mid \bar{\mathbf{z}}) = \mathcal{N}\big(\mathbf{z} \mid \mathbf{K}_{\mathbf{z}\bar{\mathbf{z}}}\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}^{-1}\bar{\mathbf{z}},\ \mathbf{K}_{\mathbf{z}\mathbf{z}} - \mathbf{K}_{\mathbf{z}\bar{\mathbf{z}}}\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}^{-1}\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}\big). \tag{4}$$

This construction generalises the inducing-point formulation and, depending on the chosen operator, can induce exploitable algebraic structure within the covariance blocks. In IDGPVAE, we apply this inter-domain construction independently to each latent channel $z^l(\cdot)$, yielding inducing variables $\bar{\mathbf{Z}} = [\bar{\mathbf{z}}^l]_{l=1}^L \in \mathbb{R}^{M\times L}$ across all latent GPs. The overall generative process remains identical to (1); the key difference lies in the richer class of inducing features available for inference.

### 3.2 Training objective and connection to inducing-point GPVAE

We now derive the training objective induced by the construction above. We augment the generative model by explicitly including the inducing variables $\bar{\mathbf{Z}}$,

$$p_{\psi,\theta}(\mathbf{Y}, \mathbf{Z}, \bar{\mathbf{Z}}) = p_\theta(\mathbf{Y} \mid \mathbf{Z})\, p_\psi(\mathbf{Z} \mid \bar{\mathbf{Z}})\, p_\psi(\bar{\mathbf{Z}}),$$

where $p_\psi(\mathbf{Z} \mid \bar{\mathbf{Z}})$ is the multi-channel version of the conditional Gaussian (4). The dependence on $\mathbf{X}$ is suppressed for brevity. Given a mini-batch $\mathcal{B}$ with $\mathbf{Y}_\mathcal{B} = [\mathbf{y}_n]_{n\in\mathcal{B}}$, $\mathbf{Z}_\mathcal{B} = [\mathbf{z}_n]_{n\in\mathcal{B}}$, the variational posterior maintains the Gaussian conditional $p(\mathbf{z}_n \mid \bar{\mathbf{Z}})$ for each latent variable $\mathbf{z}_n$:

$$q_{\psi,\varphi}(\mathbf{z}_n, \bar{\mathbf{Z}} \mid \mathbf{Y}_\mathcal{B}) = p_\psi(\mathbf{z}_n \mid \bar{\mathbf{Z}})\, q_{\psi,\varphi}(\bar{\mathbf{Z}} \mid \mathbf{Y}_\mathcal{B}) = \prod_{l=1}^{L} p_\psi(z_n^l \mid \bar{\mathbf{z}}^l)\, q_{\psi,\varphi}(\bar{\mathbf{z}}^l \mid \mathbf{Y}_\mathcal{B}). \tag{5}$$

Then, we derive the following reconstruction-minus-KL objective:

$$\mathcal{L}_{\text{IDGPVAE}} = \tfrac{N}{|\mathcal{B}|}\sum_{n\in\mathcal{B}} \mathbb{E}_{q_{\psi,\varphi}(\mathbf{z}_n\mid\mathbf{Y}_\mathcal{B})}[\log p_\theta(\mathbf{y}_n \mid \mathbf{z}_n)] - \mathrm{KL}\big[q_{\psi,\varphi}(\bar{\mathbf{Z}} \mid \mathbf{Y}_\mathcal{B}) \,\|\, p_\psi(\bar{\mathbf{Z}})\big], \tag{6}$$

where $q_{\psi,\varphi}(\mathbf{z}_n \mid \mathbf{Y}_\mathcal{B})$ is the Gaussian marginal obtained by integrating out $\bar{\mathbf{Z}}$ from (5). Here, the inducing-variable distribution $q_{\psi,\varphi}(\bar{\mathbf{z}}^l \mid \mathbf{Y}_\mathcal{B})$ denotes a variational posterior determined by the encoder outputs and the covariance blocks. Appendix A.1 provides a detailed derivation of the objective.

In principle, $q_{\psi,\varphi}(\bar{\mathbf{z}} \mid \mathbf{Y}_\mathcal{B})$ can take many reasonable parameterised forms. In this work, we use the form suggested by the pseudo-observation view in Section 2.2. For each latent channel, the encoder produces pseudo-observations $\boldsymbol{\mu}_\mathcal{B}$ and a diagonal variance matrix $\boldsymbol{\sigma}_\mathcal{B}^2$. The variational posterior in (3) then defines a heteroscedastic GP regression in the latent space. Therefore, we let $q_{\psi,\varphi}(\bar{\mathbf{z}} \mid \mathbf{Y}_\mathcal{B}) = \mathcal{N}(\bar{\mathbf{z}} \mid \mathbf{m}_\mathcal{B}, \mathbf{S}_\mathcal{B})$ follow the posterior form of sparse variational GP in Titsias (2009), and then express it in mini-batch estimation with operator-induced covariance blocks:

$$\mathbf{m}_\mathcal{B} = \tfrac{N}{|\mathcal{B}|}\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}\boldsymbol{\Sigma}_\mathcal{B}^{-1}\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}_\mathcal{B}}\boldsymbol{\sigma}_\mathcal{B}^{-2}\boldsymbol{\mu}_\mathcal{B}, \quad \mathbf{S}_\mathcal{B} = \mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}\boldsymbol{\Sigma}_\mathcal{B}^{-1}\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}, \tag{7}$$

where $\boldsymbol{\Sigma}_\mathcal{B} = \mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}} + \tfrac{N}{|\mathcal{B}|}\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}_\mathcal{B}}\boldsymbol{\sigma}_\mathcal{B}^{-2}\mathbf{K}_{\mathbf{z}_\mathcal{B}\bar{\mathbf{z}}}$. We omit latent-channel superscripts here. Since linear operators preserve the joint Gaussianity of latent and inducing variables, this parameterisation has the same functional form as SVGPVAE (Jazbec et al., 2021), allowing a controlled comparison with inducing-point counterparts (see Appendix F.3 for details). In our instantiations later, it also yields more efficient covariance structures for the chosen inducing features.

**Connection to SVGPVAE.** Unlike inducing-point formulations that obtain their objective by lower-bounding the normaliser $\mathcal{Z}_{\psi,\varphi}^l$, we derive our objective *directly* in the augmented probability space described above, which leads to a cleaner expression for training. We now show that our training objective provides an expression that generalises the commonly used SVGPVAE objective.

**Proposition 1.** *Under the variational parametrisation defined in (7), if the inducing operators $\{\mathcal{P}_m\}_{m=1}^M$ are point evaluations (i.e., Dirac delta functions) at inducing locations $\bar{\mathbf{X}}$, then the IDGPVAE objective $\mathcal{L}_{IDGPVAE}$ in (6) is algebraically equivalent to the expanded SVGPVAE objective.*

The proposition shows that our method properly generalises the conventional inducing-point construction. At the same time, the formulation above yields a simplified sparse-variational objective: $\mathcal{L}_{\text{IDGPVAE}}$ takes the standard reconstruction-minus-KL form, with all operator-specific modelling choices absorbed into the covariance blocks. Moreover, this objective can also be used for SVGPVAE, thus avoiding the need to carry an expanded variational bound for the normaliser $\mathcal{Z}_{\psi,\varphi}^l$ during training (see Appendix A.3 for the proof and a comprehensive discussion of the connection).

### 3.3 Matérn RKHS inducing features

Although our framework is operator-agnostic, both the inducing representation and its computational properties depend on the chosen operator. In this section, we focus on a family of operators defined by inner products $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ in the RKHS $\mathcal{H}$ of a half-integer Matérn kernel (Durrande et al., 2016):

$$\bar{z}_m = \mathcal{P}_m[z] \triangleq \langle z, \phi_m \rangle_{\mathcal{H}}, \qquad m = 1, 2, \cdots, M,$$

where $\{\phi_m\}_{m=1}^M$ are fixed basis functions. This inference choice is attractive for two reasons. First, it usually yields analytically tractable covariance blocks in GPVAE. Second, the resulting cross-covariance reduces to basis evaluations at the observed inputs, so it does not involve trainable inducing locations and can be precomputed once before training. Specifically, $[\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}]_{m,n} = \phi_m(\mathbf{x}_n)$ and $[\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}]_{i,j} = \langle \phi_i, \phi_j \rangle_{\mathcal{H}}$. Thus, different choices of basis family $\phi_m$ induce different representations and may yield different exploitable structures in $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}$. When the chosen basis does not yield additional exploitable structure, and the auxiliary inputs are low-dimensional, the resulting method remains in the same sparse-variational complexity regime as inducing-point GPVAE, while still avoiding cross-covariance recomputation and optimisation over inducing locations. We refer to $M$ as the *inducing scale*, since it controls the resolution of the inducing representations. For the basis families considered here, the number of learnable parameters does not grow with $M$ due to amortisation.

To investigate the efficiency and flexibility of our framework, we focus here on two representative instantiations built on recent inter-domain GPs: **(1)** Variational Fourier Features (VFFs; Hensman et al., 2018) apply sinusoidal basis functions, leading to a global spectral representation and a low-rank-plus-diagonal covariance structure. **(2)** B-Spline Features (BSFs; Cunningham et al., 2023) use compactly supported bases, inducing banded covariance matrices that reduce computation for one-dimensional inputs. These two examples illustrate how the proposed framework can accommodate different inducing biases and computational regimes for GPVAEs with low-dimensional side information. See Appendix B.2 and C for a detailed description and complexity analysis.

### 3.4 Prediction

We now describe prediction in IDGPVAE. For a test input $\mathbf{x}_*$, let $z_* = z(\mathbf{x}_*)$ denote the corresponding latent variable for a single channel. Under our construction, the predictive posterior marginalises the inter-domain inducing variables,

$$q_{\psi,\varphi}(z_* \mid \mathbf{Y}) = \int p_\psi(z_* \mid \bar{\mathbf{z}})\, q_{\psi,\varphi}(\bar{\mathbf{z}} \mid \mathbf{Y})\, \mathrm{d}\bar{\mathbf{z}},$$

where $p_\psi(z_* \mid \bar{\mathbf{z}})$ is the GP conditional induced by the chosen operators. Repeating this across latent channels gives the predictive latent vector, which is then mapped to the data space by the decoder. A straightforward prediction strategy is to construct $q_{\psi,\varphi}(\bar{\mathbf{z}} \mid \mathbf{Y})$ using all available observations. While this uses the full context, it can be memory-intensive on large datasets in practice and may oversmooth when distant observations weakly inform the target.

We therefore consider a neighbour-based alternative. For each test input $\mathbf{x}_*$, we form a local context batch $\mathcal{B}_*(\mathbf{x}_*)$ by selecting the $H$ nearest observations, and use them to compute the posterior $q_{\psi,\varphi}(\bar{\mathbf{z}} \mid \mathbf{Y}_{\mathcal{B}_*})$ instead of the full-sequence posterior. By conditioning on nearby observations, this lightweight approximation preserves local fidelity and mitigates the oversmoothing caused by full-sequence prediction when a small inducing scale yields coarse latent representations.

## 4 Related Work

**Latent-variable modelling with GP prior** Since its introduction by Casale et al. (2018), GPVAE has become a well-established line of research in structured latent-variable modelling. Subsequent work studies alternative parameterisations within the same setting (Fortuin et al., 2020; Bing et al., 2022; Pearce, 2020). Many of these early models are not scalable and are primarily used in short-sequence or full-batch settings. To improve scalability, inducing-point GPVAE variants were explored (Ashman et al., 2020; Jazbec et al., 2021). As aforementioned, SVGPVAE incorporates conventional inducing points and derives its training objective by lower-bounding the pseudo-regression normaliser. Our work keeps the generative model unchanged but extends the inference by using operator-defined inducing variables and a unified variational objective. SGPBAE (Tran et al., 2023) leverages inducing points but adopts a fully Bayesian treatment with sampling-based inference (Chen et al., 2014), which typically incurs higher computational cost. Ramchandran et al. (2021) use an inducing-point-based KL bound for longitudinal data with explicit instance or subject identifiers.

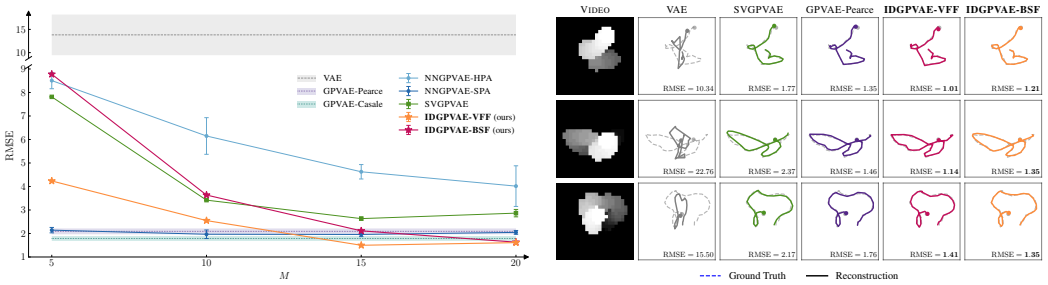
Other methods build on GPVAE by employing different inference schemes. MGPVAE (Zhu et al., 2023) performs inference via Kalman filtering and smoothing (Särkkä and Solin, 2019). Its sequential inference results in a different computational profile than inducing-variable methods. NNGPVAE (Shi et al., 2025b) uses a mean-field encoder and introduces locality during both training and prediction, whereas our neighbour-based strategy is used only at test time. Gondur et al. (2024) use a Fourier-domain parameterisation on regularly sampled multimodal data, whereas our work treats Fourier features as one instance of a broader operator-defined inference framework. Balık et al. (2025) use basis functions through Hilbert-space kernel approximation (Solin and Särkkä, 2020), leading to a weight-space formulation of the latent GPs. Instead, our model uses basis functions to define function-space inducing variables while preserving the original kernel function during inference. Chen et al. (2025) employ HiPPO-based (Gu et al., 2020) inter-domain variables in SVGPVAE for recurrent updates in online and continual learning. In contrast, we develop a general inter-domain inference framework for offline GPVAEs, investigate the accuracy–efficiency trade-offs of Fourier and B-spline features, and derive a simplified training objective.

**Sparse variational GP and inter-domain methods** SVGP and inter-domain GP methods are established tools in the GP literature (Titsias, 2009; Hensman et al., 2013, 2015; Lázaro-Gredilla and Figueiras-Vidal, 2009). Recent work on GP has explored various choices of linear operators for inter-domain constructions (Hensman et al., 2018; Dutordoir et al., 2020; Cunningham et al., 2023; Cheema and Rasmussen, 2024). Despite their success in standard GP regression and classification, their systematic integration into deep generative models via amortised inference of GPVAE remains relatively unexplored. Multi-output GPs (Álvarez et al., 2012; van der Wilk et al., 2020) extend the traditional SVGP by using techniques such as the Linear Model of Coregionalisation (LMC; Journel and Huijbregts, 2003) to capture correlations between output dimensions, but these approaches do not scale directly to high-dimensional datasets.

**Conditional, recurrent, and neural dynamical baselines** Conditional and sequential latent-variable models provide related non-GP alternatives to our method. CVAE (Sohn et al., 2015) retains factorised Gaussian priors of VAE (Kingma and Welling, 2014), but incorporates auxiliary information by feeding it directly into the encoder and decoder. Recurrent latent-variable models, such as VRNN (Chung et al., 2015) and KVAE (Fraccaro et al., 2017), are discrete-time models that capture temporal dependence through recurrent structures rather than explicit GP priors. BRITS (Cao et al., 2018) uses bidirectional recurrent dynamics for time-series imputation, but is designed as a deterministic model rather than a probabilistic generative model. Neural ODE-based models (Chen et al., 2018; Rubanova et al., 2019) represent continuous-time latent dynamics through neural differential equations and numerical integration. These methods encode dependence through neural conditioning, recurrent updates, or differential equation dynamics, while our work models correlations by leveraging the non-parametric nature offered by a GP-based function-space structure.

## 5 Experiments

We evaluate the proposed inference method through IDGPVAE-VFF and IDGPVAE-BSF on latent representation learning, data imputation, and conditional generation tasks. We compare against baselines from three groups: (i) **GP-only or VAE-only models** such as MOGP-LMC (van der Wilk




**(a)** RMSEs with different inducing scales $M$.

**(b)** Recovered latent trajectories ($M$=15).

**Figure 2:** Latent representation learning on the moving ball dataset. Colours are consistent across subplots. **(a)** Error bars and shaded regions indicate $\pm 1$ standard deviation. **(b)** The leftmost column displays original video frames overlaid with temporal shading.

et al., 2020), VAE, and HI-VAE (Nazábal et al., 2020), which test the benefit of combining GP priors with VAE-style generative mechanisms; (ii) **GPVAE-family methods** including GPVAE-Tridiag (Fortuin et al., 2020), SVGPVAE, LVAE (Ramchandran et al., 2021), SGPBAE (Tran et al., 2023), MGPVAE (Zhu et al., 2023), SA-DGBFGP (Balık et al., 2025), NNGPVAE-HPA/SPA (Shi et al., 2025b), which evaluate improvements over existing GPVAE inference schemes; (iii) **Non-GP conditional and sequential models** such as CVAE (Sohn et al., 2015), VRNN (Chung et al., 2015), BRITS (Cao et al., 2018), and LatentODE (Rubanova et al., 2019), which assess whether our approach can outperform relevant non-GP baselines. Baselines are considered on the tasks to which they are most naturally aligned. Performance is assessed using Negative Log-likelihood (NLL) and Root Mean Squared Error (RMSE). To quantify computational efficiency, we report the median wall-clock training time measured after the warm-up phase. All results are reported over five random trials with standard deviation. Appendix E provides further experimental details and additional results.

**Implementation** Our implementation is based on PyTorch (Paszke et al., 2019) and GPyTorch (Gardner et al., 2018). To exploit the banded structure induced by BSF, we implement custom C++/CUDA operators for compact banded-matrix linear algebra (e.g., Cholesky decomposition). Our goal is not to provide a general-purpose sparse linear algebra package, but to support the specific storage formats and operator patterns arising in IDGPVAE. Existing banded-matrix toolchains are either tied to legacy frameworks (Durrande et al., 2019) or are not designed for autograd workflow (Virtanen et al., 2020) and the compact banded storage (Nytko et al., 2025) in our setting. Our BSF implementation provides autograd-compatible kernels on both CPU and NVIDIA GPU, yielding notable speedups for large banded matrices compared with relying solely on native dense operators. Our IDGPVAE-VFF uses only standard PyTorch operations without custom kernels. For fair runtime comparisons, SVGPVAE in the experiments also uses the simplified objective. Further implementation details and computational analyses are given in Appendix D.

### 5.1 Moving ball: latent representation learning

Our experiments begin with the moving-ball data from Pearce (2020), in which video frames of 2D ball trajectories are sampled from a GP prior with a Matérn-$\frac{3}{2}$ kernel. The task is to infer the underlying trajectories from pixel-level frames using the latent posteriors, allowing us to assess how structured latent modelling affects the quality of the learned latent representation.

Figure 2a reports RMSE as a function of the inducing scale $M$, with shaded horizontal bands indicating the VAE and two full-batch baselines. As expected, the VAE performs worst as its factorised prior struggles to capture correlations across frames. IDGPVAE-VFF consistently outperforms inducing point-based SVGPVAE across all $M$ and matches the full-batch baselines by $M$=15. IDGPVAE-BSF exhibits a crossover behaviour: when $M$ is small, the broad support of the B-spline basis is too smooth to capture the latent variation, whereas larger $M$ provides sufficient local resolution and eventually surpasses SVGPVAE. As shown in Figure 2b, both IDGPVAE variants produce latent paths that are nearly indistinguishable from the ground truth. These results confirm that inter-domain variables can improve latent trajectory recovery over the inducing-point counterpart.

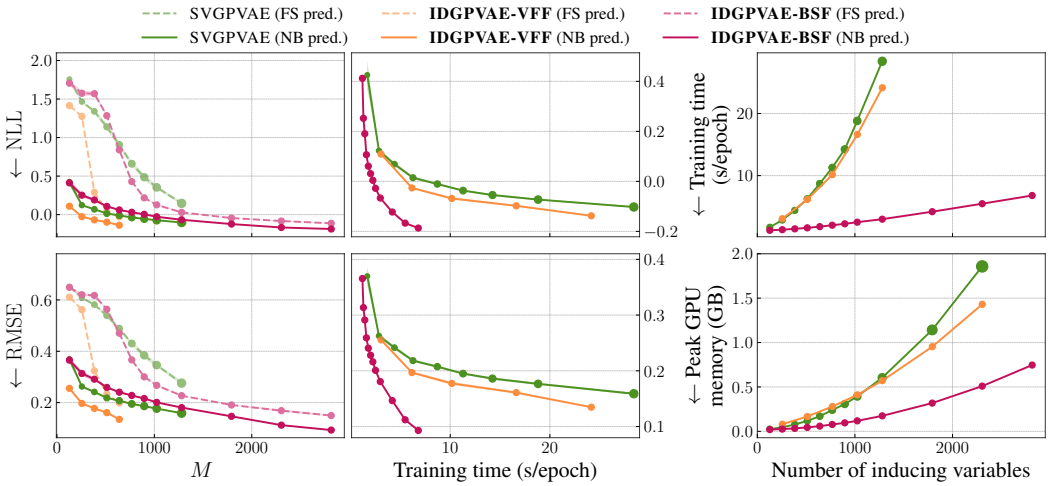



**Figure 3:** CMEMS results. Marker radius is proportional to the number of trainable parameters. Model performance in terms of NLL and RMSE versus **(left)** inducing scale $M$ under full-sequence (FS, dashed) and neighbour-based (NB, solid) prediction, versus **(middle)** wall-clock training time. **(right)** Training time and peak GPU memory versus the number of inducing variables.

### 5.2 CMEMS: accuracy–efficiency trade-off

Following the imputation setup of Tran et al. (2023), we use the CMEMS oceanic data (Lellouche et al., 2021), which comprises three years of hourly observations ($N$=26,304) with 60% of values randomly missing. This long sequence exhibits global seasonal oscillations superimposed on local high-frequency fluctuations. This imputation task is designed to evaluate whether our framework can scale effectively with inducing features while achieving a favourable accuracy–efficiency trade-off compared with existing GP latent-variable models.

**Table 1:** CMEMS imputation; SVGPVAE/IDGPVAE use ~6.2s/epoch (see Tab 13 for full results).

| MODEL | NLL (↓) | RMSE (↓) |
|---|---|---|
| SGPBAE | 4.018±0.176 | 1.352±0.075 |
| NNGPVAE-SPA | 3.213±0.442 | 1.003±0.002 |
| MOGP-LMC ($M$=1024) | 0.080±0.002 | 0.231±0.001 |
| SVGPVAE ($M$=512) | 0.015±0.006 | 0.218±0.003 |
| IDGPVAE-VFF ($M$=256) | **-0.026±0.004** | **0.197±0.003** |
| IDGPVAE-BSF ($M$=2304) | **-0.167±0.002** | **0.112±0.001** |

In Table 1, we report the best IDGPVAE and SVGPVAE across an inducing-scale sweep, subject to a per-epoch training-time budget of ~6.2s. The measured training times of the remaining baselines are no smaller than this budget. Broader sweeps over the inducing scale $M$ are shown in Figure 3 and Table 13. Without GP-style encoders, NNGPVAE and SGPBAE perform poorly in this long-sequence setting. SVGPVAE is the strongest inducing-point baseline, yet IDGPVAEs improve upon it using inter-domain formulations under a matched training budget. To further compare the effects of inter-domain and conventional inducing variables, we focus below on SVGPVAE and the two IDGPVAE instantiations.

**Inducing resolution and prediction** The left column of Figure 3 shows how predictive performance varies with the inducing scale $M$ under both FS and NB prediction, while the middle column depicts the NB results against per-epoch training time. All three inducing-variable models improve as $M$ increases. NB prediction consistently outperforms FS prediction, especially in the low-$M$ regime, where the latent representation is relatively coarse and full-sequence conditioning is more susceptible to distant, poorly reconstructed observations. As $M$ increases, this gap narrows. IDGPVAE-VFF achieves high accuracy at relatively small $M$, consistent with its global spectral inductive bias. IDGPVAE-BSF is initially less accurate at fixed $M$, but improves steadily with increasing inducing resolution and eventually achieves the best performance.

**Computational efficiency** The middle column of Figure 3 makes the accuracy–efficiency trade-off explicit: at a fixed training cost, both IDGPVAEs attain better predictive performance than SVGPVAE, with IDGPVAE-BSF tracing the most favourable frontier overall. In the right column of Figure 3, as

**Table 2:** Imputation and generation results on Rotated MNIST (see Tabs. 15/18 for full results).

| TASK | MODEL | NLL (↓) | RMSE (↓) | TIME (s/epoch ↓) |
|---|---|---|---|---|
| Corrupted-Frame Imputation | CVAE (Sohn et al., 2015) | 0.143±0.004 | 0.255±0.004 | 39.386±0.619 |
| | VRNN (Chung et al., 2015) | 0.172±0.005 | 0.284±0.004 | 50.972±2.613 |
| | BRITS (Cao et al., 2018) | — | 0.247±0.001 | 47.805±0.342 |
| | LatentODE (Rubanova et al., 2019) | 0.092±0.001 | 0.201±0.001 | 62.651±1.861 |
| | HI-VAE (Nazábal et al., 2020) | 0.097±0.001 | 0.202±0.001 | 30.583±1.104 |
| | GPVAE-Tridiag (Fortuin et al., 2020) | 0.081±0.001 | 0.187±0.001 | 216.797±2.002 |
| | SVGPVAE ($M$=5; Jazbec et al., 2021) | 0.104±0.001 | 0.215±0.001 | 35.062±0.745 |
| | MGPVAE (Zhu et al., 2023) | 0.082±0.001 | 0.187±0.001 | 80.003±2.639 |
| | NNGPVAE-HPA (Shi et al., 2025b) | 0.092±0.000 | 0.195±0.000 | 92.754±0.582 |
| | IDGPVAE-VFF ($M$=5, ours) | **0.072±0.001** | **0.173±0.001** | 33.830±1.428 |
| | IDGPVAE-BSF ($M$=10, ours) | 0.083±0.001 | **0.186±0.001** | **29.314±0.292** |
| Missing-Frame Generation | CVAE (Sohn et al., 2015) | 288.213±10.422 | 10.925±0.106 | 19.807±0.032 |
| | VRNN (Chung et al., 2015) | 82.807±4.362 | 5.984±0.155 | 84.974±0.088 |
| | BRITS (Cao et al., 2018) | — | 6.067±0.146 | 54.819±0.053 |
| | LatentODE (Rubanova et al., 2019) | 134.131±3.298 | 7.855±0.108 | 46.024±0.061 |
| | SA-DGBFGP (Balık et al., 2025) | 88.549±0.992 | 6.130±0.037 | 17.298±0.032 |
| | SVGPVAE ($M$=16; Jazbec et al., 2021) | 84.402±0.337 | 5.761±0.008 | 17.175±0.044 |
| | LVAE (Ramchandran et al., 2021) | 69.729±0.464 | 5.432±0.026 | 67.608±0.278 |
| | MGPVAE (Zhu et al., 2023) | 69.116±0.265 | 5.384±0.019 | 39.082±0.081 |
| | NNGPVAE-SPA (Shi et al., 2025b) | 74.770±0.373 | 5.558±0.013 | 36.567±0.047 |
| | IDGPVAE-VFF ($M$=8, ours) | **64.826±0.216** | **5.255±0.008** | **17.167±0.030** |
| | IDGPVAE-BSF ($M$=24, ours) | **67.792±0.095** | **5.383±0.005** | **17.081±0.020** |

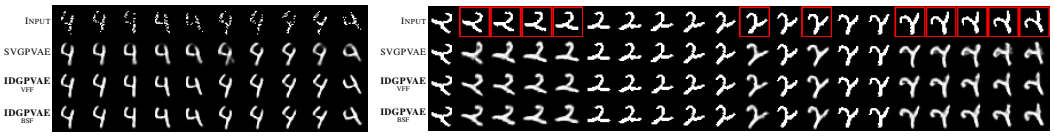



**Figure 4:** MNIST result subsets (see Figs. 11/12/14 for full results). **(left)** Corrupted-frame imputation under MCAR. **(right)** Generation of an unseen sequence; red boxes indicate missing frames.

the number of inducing variables grows, both training time and peak GPU memory increase rapidly for SVGPVAE. While remaining in the cubic regime, IDGPVAE-VFF benefits from lower practical overhead through precomputed cross-covariances and a low-rank-plus-diagonal structure. In contrast, IDGPVAE-BSF scales much more gently, owing to cheaper banded factorisations and triangular solves in compact storage. This permits substantially larger inducing resolutions within a similar time budget, thereby improving the recovery of high-frequency variation. Since both IDGPVAEs maintain a constant number of trainable parameters as $M$ increases, the improved trade-off can be attributed to the inter-domain construction rather than parameter growth.

### 5.3 Rotated MNIST: high-dimensional imputation and generation

Rotated MNIST contains many sequences of handwritten digits, each undergoing consecutive rotations over time (Krishnan et al., 2015). We consider two complementary protocols from the literature: (1) corrupted-frame imputation following Fortuin et al. (2020), where each frame is partially observed, and (2) missing-frame generation on longer videos following Zhu et al. (2023), where entire frames are absent at random timestamps. These tasks serve to evaluate how well the model learns latent dynamics and whether it provides accurate high-dimensional imputation and coherent generation. We present a subset of the results in Table 2 and Figure 4; see Appendix E.3 for additional results.

**Corrupted-frame imputation** The sequences are 10 frames long, with ~60% of pixels missing in each frame. Following Fortuin et al. (2020), we evaluate three distinct missingness mechanisms: MCAR and Temporal$^{+/-}$, where pixel absence is positively/negatively correlated across consecutive timesteps. The MCAR results are reported in Table 2. Across all three mechanisms, CVAE fails to reconstruct images faithfully, indicating that directly injecting low-dimensional auxiliary information is insufficient to match the performance achieved by placing latent GP priors in this task. VAE and HI-

VAE underperform due to their factorised priors. IDGPVAE-VFF achieves the best NLL and RMSE with competitive training time, suggesting that its global spectral representations are particularly effective here for modelling the latent rotational dynamics. While IDGPVAE-BSF requires more inducing variables to match the strongest baselines, it does so at a much lower training cost.

**Missing-frame generation** Following Zhu et al. (2023), we consider length-100 sequences where each digit completes two full rotations; each frame is independently masked with probability 0.6, leaving the remaining sequence with varying lengths. This task requires generating high-dimensional outputs on unseen sequences at unobserved timestamps. As shown in Table 2, the non-GP models considered struggle to generate data and estimate uncertainty well simultaneously. CVAE fails to produce reasonable generations, whereas VRNN, BRITS, and LatentODE require much longer training times. SA-DGBFGP also underperforms our models, possibly because encoding instance-specific latents from only the first frame is less effective for long sequences. Both IDGPVAE variants achieve lower NLL than the baselines considered, including inducing-point methods, while maintaining competitive training times. Representative examples in Figure 4 and 14 show that our models produce sharper, more coherent digits across long gaps, whereas the baselines show visible blurring at some timestamps.

## 6 Conclusion

We presented IDGPVAE, a general inference framework that integrates inter-domain approximation into deep generative models. The proposed method extends scalable GPVAE beyond conventional inducing points by defining inducing variables as linear functionals of latent GPs. It provides a simplified training objective, with the standard inducing-point GPVAE arising as a special case. In our instantiations, different operator choices yield different inducing representations and exploitable computational structures. Empirical evaluations show that our approach can achieve a more favourable accuracy–efficiency trade-off than previous counterparts.

**Limitations and future work** While the proposed framework is agnostic to feature choices, the strongest computational gains are feature-dependent rather than automatic consequences of the general framework. Our current instantiations are most effective for low-dimensional auxiliary information. Future work could pursue two directions: developing specialised inducing features for more complex input spaces (e.g., Dutordoir et al., 2020), and integrating our inference method into richer generative architectures (e.g., Chen and Li, 2023). Please see Appendix G for more discussion.

## Acknowledgments

We thank the anonymous reviewers for their valuable feedback and constructive suggestions. We also gratefully acknowledge CSF3 at the University of Manchester for providing the GPU resources used in this work. Xinxing is funded by the UoM-CSC Joint Scholarship, and Xiaoyu is supported by the UoM Departmental Studentship.

## References

Álvarez, M. A. and Lawrence, N. D. (2011). Computationally efficient convolved multiple output Gaussian processes. *Journal of Machine Learning Research*, 12(41):1459–1500.

Álvarez, M. A., Rosasco, L., and Lawrence, N. D. (2012). Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3):195–266.

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D. (1999). *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition.

Ashman, M., So, J., Tebbutt, W., Fortuin, V., Pearce, M., and Turner, R. E. (2020). Sparse Gaussian process variational autoencoders. *arXiv preprint arXiv:2010.10177*.

Balık, M. Y., Sinelnikov, M., Ong, P., and Lähdesmäki, H. (2025). Bayesian basis function approximation for scalable Gaussian process priors in deep generative models. In *International Conference on Machine Learning*.

Bhagat, S., Uppal, S., Yin, Z., and Lim, N. (2020). Disentangling multiple features in video sequences using Gaussian processes in variational autoencoders. In *European Conference on Computer Vision (ECCV)*.

Bing, S., Fortuin, V., and Rätsch, G. (2022). On disentanglement in Gaussian process variational autoencoders. In *Fourth Symposium on Advances in Approximate Bayesian Inference*.

Borovitskiy, V., Azangulov, I., Terenin, A., Mostowsky, P., Deisenroth, M., and Durrande, N. (2021). Matérn Gaussian processes on graphs. In Banerjee, A. and Fukumizu, K., editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 2593–2601. PMLR.

Bui, T. D. and Titsias, M. (2025). Sparse gaussian processes: Structured approximations and power-EP revisited. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Cao, W., Wang, D., Li, J., Zhou, H., Li, L., and Li, Y. (2018). BRITS: Bidirectional recurrent imputation for time series. *Advances in Neural Information Processing Systems*, 31.

Casale, F. P., Dalca, A., Saglietti, L., Listgarten, J., and Fusi, N. (2018). Gaussian process prior variational autoencoders. *Advances in Neural Information Processing Systems*, 31.

Cheema, T. M. and Rasmussen, C. E. (2024). Integrated variational Fourier features for fast spatial modelling with Gaussian processes. *Transactions on Machine Learning Research*.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Chen, T., Fox, E., and Guestrin, C. (2014). Stochastic gradient Hamiltonian Monte Carlo. In *International Conference on Machine Learning*, pages 1683–1691. PMLR.

Chen, W., Kiyohara, N., Zhu, H. B. H., Curran-Sebastian, J., Bhatt, S., and Li, Y. (2025). Recurrent memory for online interdomain Gaussian processes. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Chen, W. and Li, Y. (2023). Calibrating transformers via sparse Gaussian processes. In *The Eleventh International Conference on Learning Representations*.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. (2015). A recurrent latent variable model for sequential data. *Advances in Neural Information Processing Systems*, 28.

Cunningham, H. J., de Souza, D. A., Takao, S., van der Wilk, M., and Deisenroth, M. P. (2023). Actually sparse variational Gaussian processes. In Ruiz, F., Dy, J., and van de Meent, J.-W., editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 10395–10408. PMLR.

de Boor, C. (1978). *A practical guide to splines*, volume 27. Springer New York.

Durrande, N., Adam, V., Bordeaux, L., Eleftheriadis, S., and Hensman, J. (2019). Banded matrix operators for Gaussian Markov models in the automatic differentiation era. In Chaudhuri, K. and Sugiyama, M., editors, *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 2780–2789. PMLR.

Durrande, N., Hensman, J., Rattray, M., and Lawrence, N. D. (2016). Detecting periodicities with Gaussian processes. *PeerJ Computer Science*, 2:e50.

Dutordoir, V., Durrande, N., and Hensman, J. (2020). Sparse Gaussian processes with spherical harmonic features. In Hal Daumé III and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2793–2802. PMLR.

Duvenaud, D. K., Nickisch, H., and Rasmussen, C. (2011). Additive Gaussian processes. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc.

Eriksson, D., Dong, K., Lee, E., Bindel, D., and Wilson, A. G. (2018). Scaling Gaussian process regression with derivatives. *Advances in Neural Information Processing Systems*, 31.

Evans, T. W. and Nair, P. B. (2020). Quadruply stochastic Gaussian processes.

Fortuin, V., Baranchuk, D., Rätsch, G., and Mandt, S. (2020). GP-VAE: Deep probabilistic time series imputation. In *International Conference on Artificial Intelligence and Statistics*, pages 1651–1661. PMLR.

Fraccaro, M., Kamronn, S., Paquet, U., and Winther, O. (2017). A disentangled recognition and nonlinear dynamics model for unsupervised learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Gardner, J. R., Pleiss, G., Bindel, D., Weinberger, K. Q., and Wilson, A. G. (2018). GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems*, volume 31, pages 7587–7597.

Giles, M. B. (2008). Collected matrix derivative results for forward and reverse mode algorithmic differentiation. In Bischof, C. H., Bücker, H. M., Hovland, P., Naumann, U., and Utke, J., editors, *Advances in Automatic Differentiation*, pages 35–44, Berlin, Heidelberg. Springer Berlin Heidelberg.

Gondur, R., Sikandar, U. B., Schaffer, E., Aoi, M. C., and Keeley, S. L. (2024). Multi-modal Gaussian process variational autoencoders for neural and behavioral data. In *The Twelfth International Conference on Learning Representations*.

Gu, A., Dao, T., Ermon, S., Rudra, A., and Ré, C. (2020). HiPPO: recurrent memory with optimal polynomial projections. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Hamelijnck, O., Wilkinson, W., Loppi, N., Solin, A., and Damoulas, T. (2021). Spatio-temporal variational Gaussian processes. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 23621–23633. Curran Associates, Inc.

Hensman, J., Durrande, N., and Solin, A. (2018). Variational Fourier features for Gaussian processes. *Journal of Machine Learning Research*, 18(151):1–52.

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, page 282–290, Arlington, Virginia, USA. AUAI Press.

Hensman, J., Matthews, A., and Ghahramani, Z. (2015). Scalable Variational Gaussian Process Classification. In Lebanon, G. and Vishwanathan, S. V. N., editors, *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 351–360, San Diego, California, USA. PMLR.

Higgins, I., Matthey, L., Pal, A., Burgess, C. P., Glorot, X., Botvinick, M. M., Mohamed, S., and Lerchner, A. (2017). beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851.

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

Journel, A. and Huijbregts, C. (2003). *Mining Geostatistics*. Blackburn Press.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational Bayes. In Bengio, Y. and LeCun, Y., editors, *International Conference on Learning Representations*.

Krishnan, R. G., Shalit, U., and Sontag, D. (2015). Deep Kalman filters. *arXiv preprint arXiv:1511.05121*.

Lázaro-Gredilla, M. and Figueiras-Vidal, A. (2009). Inter-domain Gaussian processes for sparse inference using inducing features. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C., and Culotta, A., editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lellouche, J.-M., Greiner, E., Bourdallé-Badie, R., Garric, G., Melet, A., Drévillon, M., Bricaud, C., Hamon, M., Le Galloudec, O., Régnier, C., Candela, T., Testut, C.-E., Gasparin, F., Ruggiero, G., Benkiran, M., Drillet, Y., and Le Traon, P.-Y. (2021). The copernicus global 1/12° oceanic and sea ice glorys12 reanalysis. *Frontiers in Earth Science*, 9.

Moreta, L. S., Rønning, O., Al-Sibahi, A. S., Hein, J., Theobald, D., and Hamelryck, T. (2022). Ancestral protein sequence reconstruction using a tree-structured Ornstein-Uhlenbeck variational autoencoder. In *International Conference on Learning Representations*.

Murray, I. (2016). Differentiation of the Cholesky decomposition.

Nazábal, A., Olmos, P. M., Ghahramani, Z., and Valera, I. (2020). Handling incomplete heterogeneous data using vaes. *Pattern Recognition*, 107:107501.

Nytko, N., Taghibakhshi, A., Zaman, T. U., MacLachlan, S., Olson, L. N., and West, M. (2025). Optimized sparse matrix operations for reverse mode automatic differentiation. *SIAM Journal on Scientific Computing*, 47(5):C1115–C1143.

Padidar, M., Zhu, X., Huang, L., Gardner, J., and Bindel, D. (2021). Scaling Gaussian processes with derivative information using variational inference. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 6442–6453. Curran Associates, Inc.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pages 8024–8035.

Pearce, M. (2020). The Gaussian process prior VAE for interpretable latent dynamics from pixels. In *Symposium on Advances in Approximate Bayesian Inference*, pages 1–12. PMLR.

Prautzsch, H., Boehm, W., and Paluszny, M. (2002). *Bézier and B-spline techniques*. Springer Science & Business Media.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc.

Ramchandran, S., Tikhonov, G., Kujanpää, K., Koskinen, M., and Lähdesmäki, H. (2021). Longitudinal variational autoencoder. In *International Conference on Artificial Intelligence and Statistics*, pages 3898–3906. PMLR.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France. PMLR.

Riutort-Mayol, G., Bürkner, P.-C., Andersen, M. R., Solin, A., and Vehtari, A. (2022). Practical Hilbert space approximate Bayesian Gaussian processes for probabilistic programming. *Statistics and Computing*, 33(1):17.

Rubanova, Y., Chen, R. T., and Duvenaud, D. K. (2019). Latent ordinary differential equations for irregularly-sampled time series. *Advances in Neural Information Processing Systems*, 32.

Rudner, T. G. J., Sejdinovic, D., and Gal, Y. (2020). Inter-domain deep Gaussian processes. In Daumé, III, H. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8286–8294. PMLR.

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

Shi, X., Baldwin-McDonald, T., and Álvarez, M. A. (2025a). Adaptive RKHS Fourier features for compositional Gaussian process models. In *International Conference on Artificial Intelligence and Statistics*, pages 1738–1746. PMLR.

Shi, X., Jiang, X., and Álvarez, M. A. (2025b). Neighbour-driven Gaussian process variational autoencoders for scalable structured latent modelling. In *Forty-second International Conference on Machine Learning*.

Snelson, E. and Ghahramani, Z. (2005). Sparse Gaussian processes using pseudo-inputs. In Weiss, Y., Schölkopf, B., and Platt, J., editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press.

Sohn, K., Lee, H., and Yan, X. (2015). Learning structured output representation using deep conditional generative models. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Solin, A. and Särkkä, S. (2020). Hilbert space methods for reduced-rank Gaussian process regression. *Statistics and Computing*, 30(2):419–446.

Titsias, M. (2009). Variational learning of inducing variables in sparse Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 567–574. PMLR.

Titsias, M. (2025). New bounds for sparse variational Gaussian processes. In *Forty-second International Conference on Machine Learning*.

Tran, B.-H., Shahbaba, B., Mandt, S., and Filippone, M. (2023). Fully Bayesian autoencoders with latent sparse Gaussian processes. In *International Conference on Machine Learning*, pages 34409–34430. PMLR.

van der Wilk, M., Dutordoir, V., John, S. T., Artemev, A., Adam, V., and Hensman, J. (2020). A framework for interdomain and multioutput Gaussian processes. *arXiv preprint arXiv:2003.01115*.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

Walter, S. (2012). *Structured Higher-order Algorithmic Differentiation in the Forward and Reverse Mode with Application in Optimum Experimental Design*. PhD thesis, Humboldt University of Berlin, Faculty of Mathematics and Natural Sciences II.

Zhu, H., Balsells-Rodas, C., and Li, Y. (2023). Markovian Gaussian process variational autoencoders. In *International Conference on Machine Learning*, pages 42938–42961. PMLR.

## Appendix Contents


**Acronyms and notation** **16**

**Summary of component attribution** **17**

**A Objective derivation and equivalence** **17**

A.1 Training objective of IDGPVAE . . . . . 17

A.2 Review of SVGPVAE ELBO under pseudo-observations . . . . . 18

A.3 Equivalence of the variational objectives . . . . . 20

**B Computational complexity analysis** **21**

B.1 Explicit expressions . . . . . 21

B.2 Computational complexity . . . . . 22

**C Inducing features** **24**

C.1 Matérn family and RKHS inner product . . . . . 25

C.2 Variational Fourier feature . . . . . 25

C.3 B-spline feature . . . . . 27

**D Custom differentiable banded operators** **28**

D.1 Motivation and design requirements . . . . . 28

D.2 Compact banded storage . . . . . 28

D.3 Differentiation of matrix operations . . . . . 29

D.4 Banded Cholesky decomposition . . . . . 29

D.5 Banded triangular matrix solving . . . . . 31

D.6 Empirical efficiency . . . . . 31

**E Experimental details and additional results** **32**

E.1 Moving ball . . . . . 33

E.2 CMEMS . . . . . 34

E.3 Rotated MNIST . . . . . 36

**F SVGP and inter-domain GP with heterogeneous noise** **38**

F.1 SVGP . . . . . 39

F.2 Inter-domain GP . . . . . 40

F.3 Heterogeneous noise . . . . . 41

**G Limitations and future work** **42**

**H Licenses** **42**

## Acronyms and notation

This paper lists all acronyms and key mathematical notation in the following Table 3 and Table 4, respectively.

**Table 3:** Acronym summary.

| Category | Acronym | Meaning |
|---|---|---|
| MODELS | GP | Gaussian Process |
| | SVGP | Sparse Variational Gaussian Process |
| | MOGP-LMC | Multi-Output Gaussian Process with Linear Model of Coregionalization |
| | VAE | Variational AutoEncoder |
| | CVAE | Conditional Variational AutoEncoder |
| | GPVAE | Gaussian Process Variational AutoEncoder |
| | HI-VAE | Heterogeneous-Incomplete Variational AutoEncoder |
| | SVGPVAE | Sparse Variational Gaussian Process Variational AutoEncoder |
| | SGPBAE | Sparse Gaussian Process Bayesian AutoEncoder |
| | MGPVAE | Markovian Gaussian Process Variational AutoEncoder |
| | DGBFGP | Deep Generative Basis Function Gaussian Process |
| | LVAE | Longitudinal Variational AutoEncoder |
| | NNGPVAE | Nearest Neighbour Gaussian Process Variational AutoEncoder |
| | VFF | Variational Fourier Feature |
| | BSF | B-Spline Feature |
| | IDGPVAE | Inter-Domain Gaussian Process Variational AutoEncoder |
| CONCEPTS | NLL | Negative Log-Likelihood |
| | RMSE | Root Mean Square Error |
| | KL | Kullback-Leibler divergence |
| | ELBO | Evidence Lower BOund |
| DATASETS | MCAR | Missing Completely At Random |
| | CMEMS | Copernicus Marine Environment Monitoring Service |

**Table 4:** Overview of notation and matrix shapes.

| Symbol | Description |
|---|---|
| ARCHITECTURES | |
| $k_\psi(\mathbf{x}, \mathbf{x}')$ | GP kernel function with parameters $\psi$ |
| $\tilde{q}_\varphi(\mathbf{Z} \mid \mathbf{Y})$ | Encoder network with parameters $\varphi$ |
| $q_{\psi,\varphi}(\mathbf{Z} \mid \mathbf{Y}, \mathbf{X})$ | Variational posterior distribution of the latent GPs |
| $p_\theta(\mathbf{Y} \mid \mathbf{Z})$ | Decoder network with parameters $\theta$ |
| $M/H$ | Inducing scale / number of nearest neighbours |
| VARIABLES | |
| $\mathbf{y}_n \in \mathbb{R}^K$ | The $n^{\text{th}}$ observations (e.g., image frame or sequence value) |
| $\mathbf{Y} \in \mathbb{R}^{N \times K}$ | Observation collection |
| $\mathbf{x}_n \in \mathbb{R}^D$ | The auxiliary input of the $n^{\text{th}}$ observation (e.g., timestamp or spatial location) |
| $\mathbf{X} \in \mathbb{R}^{N \times D}$ | Auxiliary information collection |
| $z^l(\cdot)$ | Latent GP function at the $l^{\text{th}}$ latent channel |
| $\mathbf{z}^l \in \mathbb{R}^N$ | Latent random variables at the $l^{\text{th}}$ latent channel |
| $\mathbf{z}_n \in \mathbb{R}^L$ | Latent vector in the latent space; output random variables of the latent GPs |
| $\mathbf{Z} \in \mathbb{R}^{N \times L}$ | Latent variable collection; |
| $\bar{\mathbf{Z}} \in \mathbb{R}^{M \times L}$ | Inducing variables of the latent GPs |
| $\bar{\mathbf{X}} \in \mathbb{R}^{M \times D}$ | Inducing locations of the latent SVGP in SVGPVAE |
| $\boldsymbol{\mu}_n \in \mathbb{R}^L$ | Latent pseudo observations produced by the encoder for $(\mathbf{y}_n, \mathbf{x}_n)$ |
| $\boldsymbol{\sigma}_n^2 \in \mathbb{R}^L$ | Latent pseudo variance produced by the encoder for $(\mathbf{y}_n, \mathbf{x}_n)$ |
| $\boldsymbol{\mu}_\mathcal{B} \in \mathbb{R}^{\lvert\mathcal{B}\rvert}$ | Latent pseudo observations of mini-batch $\mathcal{B}$ (at the latent channel $l$) |
| $\boldsymbol{\sigma}_\mathcal{B}^2 \in \mathbb{R}^{\lvert\mathcal{B}\rvert \times \lvert\mathcal{B}\rvert}$ | Diagonal pseudo variance matrix of mini-batch $\mathcal{B}$ (at the latent channel $l$) |

## Summary of component attribution

The proposed inference method contains several interacting components. Its main benefit is best understood as a coupled representation–computation effect: operator-defined inducing variables change the variational representation, while the resulting covariance structure determines which inducing resolutions are computationally feasible. As the framework couples these two aspects, complete separation is neither natural nor always meaningful. Nevertheless, the experiments provide several diagnostics for their roles, which we summarise below.

**Table 5:** Summary of component contributions and empirical evidence

| Component | Primary Contribution | Main Empirical Evidence |
|---|---|---|
| Basis choice (BSF) | Banded structure: further reduces computational cost and memory usage; allows more inducing variables for latent approximation. | Figure 3,8,9, Table 13; Table 15,18, Figure 11,12,14. |
| Basis choice (VFF) | Spectral representation: effectively captures global spectral patterns with relatively few features. | Figure 2; Figure 3(left),8,9, Table 13; Table 18,20, Figure 14. |
| Inducing scale | Latent resolution control: increasing $M$ generally improves the fidelity of the latent GP approximation. | Figure 2a, Table 11; Figure 3(left),9, Table 13; Table 16,20,21. |
| Neighbour-based prediction | Local fidelity: practical for large datasets; mitigates the influence of distant observations when $M$ is small. | Figure 3(left),10; Table 21, Figure 13. |

# A Objective derivation and equivalence

This section derives the IDGPVAE training objective (Appendix A.1), reviews the SVGPVAE derivation from the pseudo-observation formulation (Appendix A.2), and establishes the connection between the two objectives (Appendix A.3).

## A.1 Training objective of IDGPVAE

Given $N$ data points, GPVAE considers three categories of variables

$$\mathbf{Y} = [\mathbf{y}_n]_{n=1}^N \in \mathbb{R}^{N\times K}, \quad \mathbf{X} = [\mathbf{x}_n]_{n=1}^N \in \mathbb{R}^{N\times D}, \quad \mathbf{Z} = [\mathbf{z}_n]_{n=1}^N \in \mathbb{R}^{N\times L},$$

where $\mathbf{y}_n \in \mathbb{R}^K$, $\mathbf{x}_n \in \mathbb{R}^D$, $\mathbf{z}_n \in \mathbb{R}^L$ denote the $n^{\text{th}}$ observation, auxiliary information, and latent variable, respectively. $\mathbf{z}^l \in \mathbb{R}^N$ represents the random vector at the $l^{\text{th}}$ column (channel) of $\mathbf{Z}$.

We derive the training objective in a manner analogous to SVGP (Hensman et al., 2015). As in the main text, the joint generative process is given by

$$p_{\psi,\theta}(\mathbf{Y}, \mathbf{Z}, \bar{\mathbf{Z}}) = p_\theta(\mathbf{Y} \mid \mathbf{Z})\, p_\psi(\mathbf{Z} \mid \bar{\mathbf{Z}})\, p_\psi(\bar{\mathbf{Z}}),$$

where $\bar{\mathbf{Z}}$ denotes a collection of inter-domain inducing variables, and $p_\psi(\mathbf{Z} \mid \bar{\mathbf{Z}})$ is a Gaussian conditional distribution. Inspired by SVGP, we consider a general variational posterior that preserves the Gaussian conditional and factors as

$$q_{\psi,\varphi}(\mathbf{Z}, \bar{\mathbf{Z}}) = p_\psi(\mathbf{Z} \mid \bar{\mathbf{Z}}) q_{\psi,\varphi}(\bar{\mathbf{Z}}),$$

where the variational posterior over the inducing variables is modelled by the encoder output with parameters $\varphi$ and covariance matrices with parameters $\psi$. Applying Jensen's inequality to the joint

distribution gives

$$\begin{aligned}\log p(\mathbf{Y}) &\geqslant \int q_{\psi,\varphi}(\mathbf{Z},\bar{\mathbf{Z}}) \log \frac{p_{\psi,\theta}(\mathbf{Y},\mathbf{Z},\bar{\mathbf{Z}})}{q_{\psi,\varphi}(\mathbf{Z},\bar{\mathbf{Z}})}\,\mathrm{d}\bar{\mathbf{Z}}\,\mathrm{d}\mathbf{Z} \\ &= \int q_{\psi,\varphi}(\mathbf{Z},\bar{\mathbf{Z}}) \log \frac{p_\theta(\mathbf{Y}\mid\mathbf{Z})\cancel{p_\psi(\mathbf{Z}\mid\bar{\mathbf{Z}})}p_\psi(\bar{\mathbf{Z}})}{\cancel{p_\psi(\mathbf{Z}\mid\bar{\mathbf{Z}})}q_{\psi,\varphi}(\bar{\mathbf{Z}})}\,\mathrm{d}\bar{\mathbf{Z}}\,\mathrm{d}\mathbf{Z} \\ &= \int q_{\psi,\varphi}(\mathbf{Z},\bar{\mathbf{Z}}) \log p_\theta(\mathbf{Y}\mid\mathbf{Z})\,\mathrm{d}\bar{\mathbf{Z}}\,\mathrm{d}\mathbf{Z} - \mathrm{KL}\left[q_{\psi,\varphi}(\bar{\mathbf{Z}}) \,||\, p_\psi(\bar{\mathbf{Z}})\right].\end{aligned}$$

Given a mini-batch $\mathcal{B}$ with $\mathbf{Y}_\mathcal{B}=[\mathbf{y}_n]_{n\in\mathcal{B}}$, $\mathbf{Z}_\mathcal{B}=[\mathbf{z}_n]_{n\in\mathcal{B}}$, the model parameterises $q_{\psi,\varphi}(\bar{\mathbf{Z}})$ as a function of $\mathbf{Y}_\mathcal{B}$ for amortisation:

$$q_{\psi,\varphi}(\bar{\mathbf{Z}}\mid\mathbf{Y}_\mathcal{B}) = \prod_{l=1}^{L} q_{\psi,\varphi}(\bar{\mathbf{z}}^l\mid\mathbf{Y}_\mathcal{B}) = \prod_{l=1}^{L}\mathcal{N}(\bar{\mathbf{z}}^l\mid\mathbf{m}_\mathcal{B}^l,\mathbf{S}_\mathcal{B}^l),$$

where $\mathbf{m}_\mathcal{B}$ and $\mathbf{S}_\mathcal{B}$ are the encoder outputs for the mini-batch (see, for instance, Eq.(7)). This induces an amortised dependence of the associated latent variables $\mathbf{Z}_\mathcal{B}$ on the corresponding mini-batch. Consequently, we can rewrite $q_{\psi,\varphi}(\mathbf{Z},\bar{\mathbf{Z}})$ in a mini-batch form as

$$q_{\psi,\varphi}(\mathbf{Z}_\mathcal{B},\bar{\mathbf{Z}}\mid\mathbf{Y}_\mathcal{B}) = p_\psi(\mathbf{Z}_\mathcal{B}\mid\bar{\mathbf{Z}})q_{\psi,\varphi}(\bar{\mathbf{Z}}\mid\mathbf{Y}_\mathcal{B}), \tag{8}$$

where $p_\psi(\mathbf{Z}_\mathcal{B}\mid\bar{\mathbf{Z}})$ denotes the GP conditional over the latent variables of the mini-batch.

The training objective for IDGPVAE is then obtained by a mini-batch approximation for the expected log-likelihood term under Gaussian marginals, which can be written as

$$\begin{aligned}\mathcal{L}_{\text{IDGPVAE}} &= \tfrac{N}{|\mathcal{B}|}\mathbb{E}_{q_{\psi,\varphi}(\mathbf{Z}_\mathcal{B}\mid\mathbf{Y}_\mathcal{B})}[\log p_\theta(\mathbf{Y}_\mathcal{B}\mid\mathbf{Z}_\mathcal{B})] - \mathrm{KL}\big[q_{\psi,\varphi}(\bar{\mathbf{Z}}\mid\mathbf{Y}_\mathcal{B})\,\|\,p_\psi(\bar{\mathbf{Z}})\big] \\ &= \tfrac{N}{|\mathcal{B}|}\sum_{n\in\mathcal{B}}\mathbb{E}_{q_{\psi,\varphi}(\mathbf{z}_n\mid\mathbf{Y}_\mathcal{B})}[\log p_\theta(\mathbf{y}_n\mid\mathbf{z}_n)] - \mathrm{KL}\big[q_{\psi,\varphi}(\bar{\mathbf{Z}}\mid\mathbf{Y}_\mathcal{B})\,\|\,p_\psi(\bar{\mathbf{Z}})\big],\end{aligned} \tag{9}$$

where $q_{\psi,\varphi}(\mathbf{z}_n\mid\mathbf{Y}_\mathcal{B})$ is the $n^{\text{th}}$ marginal distribution of $q_{\psi,\varphi}(\mathbf{Z}_\mathcal{B},\bar{\mathbf{Z}}\mid\mathbf{Y}_\mathcal{B})$, given by

$$q_{\psi,\varphi}(\mathbf{z}_n\mid\mathbf{Y}_\mathcal{B}) = \int p_\psi(\mathbf{z}_n\mid\bar{\mathbf{Z}})q_{\psi,\varphi}(\bar{\mathbf{Z}}\mid\mathbf{Y}_\mathcal{B})\,\mathrm{d}\bar{\mathbf{Z}} = \prod_{l=1}^{L}\mathcal{N}(z_n^l\mid m_s(\mathbf{x}_n),\Sigma_s(\mathbf{x}_n)).$$

The posterior mean $m_s$ and variance $\Sigma_s$ above are expressed as

$$\begin{cases} m_s(\mathbf{x}_n) &= \mathbf{k}_{z_n\bar{\mathbf{z}}}\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}^{-1}\mathbf{m}_\mathcal{B}, \\ \Sigma_s(\mathbf{x}_n) &= k_{z_nz_n} + \mathbf{k}_{z_n\bar{\mathbf{z}}}\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}^{-1}\left[\mathbf{S}_\mathcal{B} - \mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}\right]\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}^{-1}\mathbf{k}_{\bar{\mathbf{z}}z_n}, \end{cases} \tag{10}$$

where we suppress the superscript $l$ for latent channels for brevity.

The training objective of IDGPVAE has a standard VAE form, with an expected log-likelihood term and a KL divergence term. We further provide closed-form expressions and computational analysis of the objective $\mathcal{L}_{\text{IDGPVAE}}$ in Appendix B.1 and Appendix B.2, respectively.

### A.2 Review of SVGPVAE ELBO under pseudo-observations

The original objective derivation of SVGPVAE starts from Pearce (2020).

**Pearce's ELBO** To approximate the true posterior distribution

$$p(\mathbf{Z}\mid\mathbf{Y}) \propto p_\psi(\mathbf{Z})p_\theta(\mathbf{Y}\mid\mathbf{Z}),$$

a straightforward approach is to use an encoder network that directly outputs the mean and variance of a variational distribution. Pearce (2020) designs a different formulation for the variational distribution by replacing the likelihood $p_\theta(\mathbf{Y}\mid\mathbf{Z})$ with the encoder network $\tilde{q}_\varphi(z_n^l\mid\mathbf{y}_n)=\mathcal{N}(z_n^l\mid\mu_n^l(\mathbf{y}_n),\sigma_n^{2,l}(\mathbf{y}_n))$,

$$q_{\psi,\varphi}(\mathbf{Z}\mid\mathbf{Y},\mathbf{X}) = \prod_{l=1}^{L}\frac{p_\psi(\mathbf{z}^l\mid\mathbf{X})\prod_{n=1}^{N}\tilde{q}_\varphi(z_n^l\mid\mathbf{y}_n)}{\mathcal{Z}_{\psi,\varphi}^l(\mathbf{Y},\mathbf{X})}, \tag{11}$$

where $\mathcal{Z}^l_{\psi,\varphi}$ is the normalising constant w.r.t. $\mathbf{z}^l$. We collect the outputs of the encoder network as

$$\boldsymbol{\mu} = [\mu_n^l]_{n=1,l=1}^{N,L} \in \mathbb{R}^{N\times L}, \quad \boldsymbol{\sigma} = [\sigma_n^l]_{n=1,l=1}^{N,L} \in \mathbb{R}^{N\times L}.$$

Then, the ELBO $\mathcal{L}_P$ proposed in Pearce (2020) can be written as

$$\begin{aligned}\mathcal{L}_P &= \int q_{\psi,\varphi}(\mathbf{Z} \mid \mathbf{Y}, \mathbf{X}) \log \frac{p_\theta(\mathbf{Y} \mid \mathbf{Z}) p_\psi(\mathbf{Z} \mid \mathbf{X})}{q_{\psi,\varphi}(\mathbf{Z} \mid \mathbf{Y}, \mathbf{X})} \,\mathrm{d}\mathbf{Z} \\ &= \sum_{n=1}^{N} \mathbb{E}_{q_{\psi,\varphi}(\mathbf{z}_n)} \left[\log p_\theta(\mathbf{y}_n \mid \mathbf{z}_n)\right] - \sum_{l=1}^{L}\sum_{n=1}^{N} \mathbb{E}_{q_{\psi,\varphi}(z_n^l)} \left[\log \tilde{q}_\varphi(z_n^l \mid \mathbf{y}_n)\right] \\ &\quad + \sum_{l=1}^{L} \log \mathcal{Z}^l_{\psi,\varphi}(\mathbf{Y}, \mathbf{X}).\end{aligned} \tag{12}$$

In this derivation, the prior distribution $p_\psi(\mathbf{z}^l \mid \mathbf{X})$ cancels out, and in the final equality we omit the dependence on $\mathbf{X}$ in $q_{\psi,\varphi}$ for notational simplicity.

**Latent GP regression** For each latent channel $l$, the input-target pairs $(\mathbf{X}, \boldsymbol{\mu}^l)$ and heteroscedastic noise variances $\boldsymbol{\sigma}^{2,l}$ define a standard GP regression problem whose marginal likelihood is $\mathcal{Z}^l_{\psi,\varphi}(\mathbf{Y}, \mathbf{X})$, and the corresponding posterior is $q_{\psi,\varphi}$ in (11).[2] In other words, given $\{\mathbf{X}, \mathbf{Y}\}$, the inference network $\tilde{q}_\varphi$ produces corresponding *pseudo*-observations and variances for the latent GP regression. Therefore, the normaliser is given by

$$\begin{aligned}\log \mathcal{Z}^l_{\psi,\varphi}(\mathbf{Y}, \mathbf{X}) &= \log \int p_\psi(\mathbf{z}^l \mid \mathbf{X}) \prod_{n=1}^{N} \tilde{q}_\varphi(z_n^l \mid \mathbf{y}_n) \,\mathrm{d}\mathbf{z}^l \\ &= \log \mathcal{N}(\boldsymbol{\mu}^l \mid \mathbf{0}, k^l_\psi(\mathbf{X}, \mathbf{X}) + \boldsymbol{\sigma}^{2,l}),\end{aligned}$$

where $\boldsymbol{\sigma}^{2,l} = \mathrm{diag}([\sigma_1^{2,l}, \cdots, \sigma_N^{2,l}])$. While the variational distribution (11) integrates the auxiliary information $\mathbf{X}$, computing the normaliser $\log \mathcal{Z}^l_{\psi,\varphi}$ above requires $\mathcal{O}(N^3)$ GP computations and ties training to the full dataset, thereby making $\mathcal{L}_P$ unsuitable for large $N$.

**SVGPVAE objective via normaliser lower bound** As $\log \mathcal{Z}^l_{\psi,\varphi}(\mathbf{Y}, \mathbf{X})$ can be interpreted as the exact GP marginal log-likelihood in the pseudo-observation view, the *uncollapsed* SVGP bound (Hensman et al., 2013) can be applied to obtain a lower bound on this term. To this end, Jazbec et al. (2021) introduce $M(\ll N)$ inducing variables $\bar{\mathbf{Z}} = \left[\bar{\mathbf{z}}^l\right]_{l=1}^{L} \in \mathbb{R}^{M\times L}$ at associated inducing locations $\bar{\mathbf{X}} \in \mathbb{R}^{M\times D}$ (for simplicity, the inducing locations are shared across latent dimensions). We first consider a single latent channel and omit the channel superscript $l$, since the channels are independent by construction.

Let the variational distribution over the inducing points be $q(\bar{\mathbf{z}}) = \mathcal{N}(\bar{\mathbf{z}} \mid \mathbf{m}, \mathbf{S})$ and the prior be $p(\bar{\mathbf{z}}) = \mathcal{N}(\bar{\mathbf{z}} \mid \mathbf{0}, \mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}})$. The lower bound of $\log \mathcal{Z}_{\psi,\varphi}(\mathbf{Y}, \mathbf{X})$ is given by the *uncollapsed* SVGP bound $\mathcal{L}_H$ (see Eq. (4) in Hensman et al., 2013):

$$\begin{aligned}\log \mathcal{Z}_{\psi,\varphi} &\geqslant \sum_{n=1}^{N} \left\{ \log \mathcal{N}(\mu_n \mid \mathbf{k}_n^\top \mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}^{-1} \mathbf{m}, \sigma_n^2) - \frac{1}{2\sigma_n^2}\left(\tilde{k}_{nn} + \mathrm{tr}(\mathbf{S}\boldsymbol{\Lambda}_n)\right) \right\} - \mathrm{KL}[q(\bar{\mathbf{z}}) || p(\bar{\mathbf{z}})] \\ &:= \mathcal{L}_H,\end{aligned} \tag{13}$$

where $\mathbf{k}_n$ is the $n^{\text{th}}$ column of $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}$, $\tilde{k}_{nn} = k(\mathbf{x}_n, \mathbf{x}_n) - \mathbf{k}_n^\top \mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}^{-1} \mathbf{k}_n$, and $\boldsymbol{\Lambda}_n = \mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}^{-1} \mathbf{k}_n \mathbf{k}_n^\top \mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}^{-1}$. As aforementioned, $\mu_n$ and $\sigma_n^2$ play the role of observations and noise variances as in a standard GP regression. $\mathcal{L}_H$ is written as a sum of $n$ terms and can be estimated using mini-batches.

Similar to the variational family of SVGP, the joint variational posterior of the latent variables $\mathbf{z}$ and the inducing variables $\bar{\mathbf{z}}$ takes the form

$$q_s(\mathbf{z}, \bar{\mathbf{z}}) = p(\mathbf{z} \mid \bar{\mathbf{z}}) q(\bar{\mathbf{z}}),$$

[2] One can also compute the terms in the numerator using the identity of the product of two Gaussian distributions.

where $p(\mathbf{z} \mid \bar{\mathbf{z}})$ is a standard conditional Gaussian distribution. The marginal distribution over $\mathbf{z}$ is given by

$$q_s(\mathbf{z}) = \int p(\mathbf{z} \mid \bar{\mathbf{z}}) q(\bar{\mathbf{z}}) \, \mathrm{d}\bar{\mathbf{z}}.$$

Consequently, the SVGPVAE training objective is obtained by substituting $q_s(\cdot)$ and $\mathcal{L}_H$ for $q_{\psi,\varphi}(\cdot)$ and $\log \mathcal{Z}_{\psi,\varphi}$, respectively, in $\mathcal{L}_P$ (12) (also see Eq. (10) in Jazbec et al., 2021):

$$\mathcal{L}_{\text{SVGPVAE}} = \sum_{n=1}^{N} \mathbb{E}_{q_s(\mathbf{z}_n)} \left[\log p_\theta(\mathbf{y}_n \mid \mathbf{z}_n)\right] - \sum_{n=1}^{N} \sum_{l=1}^{L} \mathbb{E}_{q_s(z_n^l)} \left[\log \tilde{q}_\varphi(z_n^l \mid \mathbf{y}_n)\right] + \sum_{l=1}^{L} \mathcal{L}_H^l, \quad (14)$$

$$\approx \tfrac{N}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \mathbb{E}_{q_s(\mathbf{z}_n)} \left[\log p_\theta(\mathbf{y}_n \mid \mathbf{z}_n)\right] - \tfrac{N}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \sum_{l=1}^{L} \mathbb{E}_{q_s(z_n^l)} \left[\log \tilde{q}_\varphi(z_n^l \mid \mathbf{y}_n)\right] + \sum_{l=1}^{L} \mathcal{L}_H^l, \quad (15)$$

where $\mathcal{L}_H^l$ is the SVGP bound of the latent channel $l$ given by (13).

### A.3 Equivalence of the variational objectives

We now consider the special case where the inter-domain operators in IDGPVAE reduce to point evaluations. We adopt the SVGP-style parametrisation in (7). Under these conditions, we show the equivalence of $\mathcal{L}_{\text{IDGPVAE}}$ and $\mathcal{L}_{\text{SVGPVAE}}$. The key observation is that the summands in $\mathcal{L}_H$ cancel the cross-entropy term in $\mathcal{L}_{\text{SVGPVAE}}$ above.

**Proposition 1** (Objective equivalence)**.** *Under the variational parameterisation in (7), $\mathcal{L}_{\text{IDGPVAE}}$ (9) is algebraically equivalent to $\mathcal{L}_{\text{SVGPVAE}}$ (15) when the inducing features $\mathcal{P}_m$ reduce to point evaluations (i.e., Dirac delta functions) at inducing locations $\bar{\mathbf{X}}$.*

*Proof.* We omit the superscript $l$ for brevity.

Recall that the SVGP bound in Hensman et al. (2013) is a special case of the more general SVGP bound in Hensman et al. (2015). In the latter, the expected log-likelihood term is typically written as $\mathbb{E}_{q(f)}[\log p(y \mid f)]$ in the GP literature, and the former corresponds to the specific case of a Gaussian likelihood for $p(y \mid f)$. In our case, the summand of $\mathcal{L}_H$ in (13) therefore satisfies

$$\log \mathcal{N}\big(\mu_n \mid \mathbf{k}_n^\top \mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}^{-1} \mathbf{m}, \sigma_n^2\big) - \frac{1}{2\sigma_n^2}\Big(\tilde{k}_{nn} + \operatorname{tr}(\mathbf{S}\boldsymbol{\Lambda}_n)\Big) = \mathbb{E}_{q_s(z_n)}\big[\log \mathcal{N}(\mu_n \mid z_n, \sigma_n^2)\big].$$

By the symmetry of the Gaussian log-density in its arguments, we have

$$\mathbb{E}_{q_s(z_n)}\big[\log \tilde{q}_\varphi(z_n \mid \mathbf{y}_n)\big] = \mathbb{E}_{q_s(z_n)}\big[\log \mathcal{N}(z_n \mid \mu_n, \sigma_n^2)\big] = \mathbb{E}_{q_s(z_n)}\big[\log \mathcal{N}(\mu_n \mid z_n, \sigma_n^2)\big].$$

Thus, these two terms in (14) *cancel*, and $\mathcal{L}_{\text{SVGPVAE}}$ reduces to

$$\begin{aligned}\mathcal{L}_{\text{SVGPVAE}} &= \sum_{n=1}^{N} \mathbb{E}_{q_s(\mathbf{z}_n)} \left[\log p_\theta(\mathbf{y}_n \mid \mathbf{z}_n)\right] - \mathrm{KL}[q(\bar{\mathbf{z}}) \| p(\bar{\mathbf{z}})] \\ &\approx \tfrac{N}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \mathbb{E}_{q_s(\mathbf{z}_n)} \left[\log p_\theta(\mathbf{y}_n \mid \mathbf{z}_n)\right] - \mathrm{KL}[q(\bar{\mathbf{z}}) \| p(\bar{\mathbf{z}})].\end{aligned}$$

Note that now $\mathbf{m}$ and $\mathbf{S}$ of $q(\bar{\mathbf{z}})$ take a similar form to (7), which is determined by mini-batches $\mathcal{B}$. The parameterisation of $\mathbf{m}_\mathcal{B}$ and $\mathbf{S}_\mathcal{B}$ is inspired by the optimal variational distribution in the context of SVGP with Gaussian likelihood (see Appendix F.1 for details).

Consider the case where the basis functions become Dirac delta functions at the inducing locations $\bar{\mathbf{x}}_m$ of SVGPVAE, i.e., $\phi_m(\mathbf{x}) = \delta(\mathbf{x} - \bar{\mathbf{x}}_m)$. The corresponding inter-domain inducing variable $\bar{\mathbf{z}}_m$ then reduces to a point-evaluated inducing variable:

$$\bar{\mathbf{z}}_m = \mathcal{P}_m[z] = \int z(\mathbf{x}) \delta(\mathbf{x} - \bar{\mathbf{x}}_m) \, \mathrm{d}\mathbf{x} = z(\bar{\mathbf{x}}_m),$$

where $z(\cdot) \sim \mathcal{GP}(0, k(\cdot,\cdot'))$. Consequently, the entries of the inter-domain covariance matrices $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}$ and $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}$ reduce to standard kernel evaluations, i.e., $[k(\bar{\mathbf{X}}, \mathbf{X})]_{i,j}$ and $[k(\bar{\mathbf{X}}, \bar{\mathbf{X}})]_{i,j}$, and $\mathbf{m}_\mathcal{B}$ and $\mathbf{S}_\mathcal{B}$ in IDGPVAE take the same functional form as in SVGPVAE. Hence, for the same batch of data points, the marginal posteriors coincide when $\phi_m(\mathbf{x}) = \delta(\mathbf{x} - \bar{\mathbf{x}}_m)$, i.e., $q_{\psi,\varphi}(\mathbf{z}_n|\mathbf{Y}_\mathcal{B}) = q_s(\mathbf{z}_n)$, so the expected likelihood terms in $\mathcal{L}_{\text{IDGPVAE}}$ and $\mathcal{L}_{\text{SVGPVAE}}$ are identical. Similarly, the KL term in $\mathcal{L}_{\text{IDGPVAE}}$ reduces to that in $\mathcal{L}_{\text{SVGPVAE}}$. Therefore, we conclude that $\mathcal{L}_{\text{IDGPVAE}} = \mathcal{L}_{\text{SVGPVAE}}$ under the conditions stated in the proposition. □

**Discussion and practical implications** Proposition 1 clarifies the relationship between our objective and that of SVGPVAE. When inducing operators reduce to point evaluations, the two objectives coincide, and the SVGPVAE training objective can be written in a standard VAE style. In particular, under the Gaussian pseudo-observation factors $\tilde{q}_\varphi(z_n \mid \mathbf{y}_n)$, the expected "site" term in the pseudo-observation ELBO ($\mathcal{L}_{\text{SVGPVAE}}$) matches the per-datapoint expected log-likelihood term arising from the Gaussian-likelihood SVGP bound ($\mathcal{L}_H$); these contributions cancel algebraically.

This observation is not presented as a standalone simplification of the final objective in Jazbec et al. (2021) (see their intermediate steps that bound the normaliser). Our formulation makes this equivalence explicit and yields a cleaner objective that is easier to implement and evaluate. It also provides a direct pathway to the inter-domain generalisation: IDGPVAE is obtained by replacing the point-evaluation covariance blocks with their operator-projected counterparts.

In exact arithmetic, one may evaluate SVGPVAE either in its original expanded form or in the simplified VAE form above. The two are algebraically equivalent and therefore induce the same gradients. The simplified form avoids computing redundant terms, thereby reducing overhead. In finite-precision arithmetic, however, the expanded objective contains analytically cancelling terms, and the additional evaluations can introduce small numerical discrepancies due to rounding and cancellation. We thus use the simplified training objective for all SVGPVAEs in the experiments, thereby reducing overhead and improving numerical consistency. Hence, the reported wall-clock differences more directly reflect the effect of introducing inter-domain inducing features and their associated linear-algebra structure.

## B Computational complexity analysis

In this section, we first provide explicit expressions of the objective $\mathcal{L}_{\text{IDGPVAE}}$ and then analyse the computational complexity in detail. Throughout, we suppress the channel superscript $l$ for clarity.

### B.1 Explicit expressions

As aforementioned, the training objective of our IDGPVAE is given by

$$\mathcal{L}_{\text{IDGPVAE}} = \tfrac{N}{|\mathcal{B}|} \sum_{n\in\mathcal{B}} \mathbb{E}_{q_{\psi,\varphi}(\mathbf{z}_n|\mathbf{Y}_\mathcal{B})}[\log p_\theta(\mathbf{y}_n \mid \mathbf{z}_n)] - \mathrm{KL}\big[q_{\psi,\varphi}(\bar{\mathbf{Z}} \mid \mathbf{Y}_\mathcal{B}) \,\|\, p_\psi(\bar{\mathbf{Z}})\big]\,,$$

Given pseudo-observations $\boldsymbol{\mu}_\mathcal{B}$, pseudo-variances $\boldsymbol{\sigma}^2_\mathcal{B}$, and the covariance update $\boldsymbol{\Sigma}_\mathcal{B}$, we use the following form for the variational posterior over the inducing variables:

$$q_{\psi,\varphi}(\bar{\mathbf{z}} \mid \mathbf{Y}_\mathcal{B}) = \mathcal{N}(\bar{\mathbf{z}} \mid \mathbf{m}_\mathcal{B}, \mathbf{S}_\mathcal{B}), \quad \text{where} \quad \begin{cases} \mathbf{m}_\mathcal{B} &= \tfrac{N}{|\mathcal{B}|}\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}} \underbrace{\boldsymbol{\Sigma}_\mathcal{B}^{-1}\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}_\mathcal{B}}\boldsymbol{\sigma}_\mathcal{B}^{-2}\boldsymbol{\mu}_\mathcal{B}}_{\bar{\mathbf{m}}_\mathcal{B}}, \\ \mathbf{S}_\mathcal{B} &= \mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}\boldsymbol{\Sigma}_\mathcal{B}^{-1}\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}. \end{cases}$$

**Expected log-likelihood** Through the conditional $p_\psi(\mathbf{z} \mid \bar{\mathbf{z}})$, the marginal variational posterior over the $n^{\text{th}}$ latent variable is obtained by

$$q_{\psi,\varphi}(z_n \mid \mathbf{Y}_\mathcal{B}) = \int p_\psi(z_n \mid \bar{\mathbf{z}}) q_{\psi,\varphi}(\bar{\mathbf{z}} \mid \mathbf{Y}_\mathcal{B})\, \mathrm{d}\bar{\mathbf{z}} = \mathcal{N}\left(z_n \mid m_s(\mathbf{x}_n), \Sigma_s(\mathbf{x}_n)\right),$$

$$\text{with} \quad \begin{cases} m_s(\mathbf{x}_n) &= \tfrac{N}{|\mathcal{B}|}\mathbf{K}_{z_n\bar{\mathbf{z}}}\bar{\mathbf{m}}_\mathcal{B}, \\ \Sigma_s(\mathbf{x}_n) &= k_{z_n z_n} + \mathbf{K}_{z_n\bar{\mathbf{z}}}\left(\boldsymbol{\Sigma}_\mathcal{B}^{-1} - \mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}^{-1}\right)\mathbf{K}_{\bar{\mathbf{z}}z_n}. \end{cases}$$

Then, the expected log-likelihood term in the objective can be computed using the reparameterisation trick (Kingma and Welling, 2014): $z_n = m_s(\mathbf{x}_n) + \sqrt{\Sigma_s(\mathbf{x}_n)}\,\epsilon$, $\epsilon \sim \mathcal{N}(0,1)$.

**KL divergence** The KL term in the objective can be written in closed form as

$$\begin{aligned}
\mathrm{KL}[q_{\psi,\varphi}(\bar{\mathbf{z}}) \,\|\, p_\psi(\bar{\mathbf{z}})] &= \mathrm{KL}[\mathcal{N}(\bar{\mathbf{z}} \mid \mathbf{m}_\mathcal{B}, \mathbf{S}_\mathcal{B}) \| \mathcal{N}(\bar{\mathbf{z}} \mid \mathbf{0}, \mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}})] \\
&= \frac{1}{2}\left[\mathbf{m}_\mathcal{B}^\top \mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}^{-1} \mathbf{m}_\mathcal{B} + \mathrm{tr}\left\{\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}^{-1}\mathbf{S}_\mathcal{B}\right\} + \log\frac{|\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}|}{|\mathbf{S}_\mathcal{B}|} - M\right] \\
&= \frac{1}{2}\Bigg[\underbrace{\tfrac{N}{|\mathcal{B}|}\boldsymbol{\mu}_\mathcal{B}^\top \boldsymbol{\sigma}_\mathcal{B}^{-2}\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}_\mathcal{B}}\boldsymbol{\Sigma}_\mathcal{B}^{-1}}_{\bar{\mathbf{m}}_\mathcal{B}^\top} \cdot \underbrace{\tfrac{N}{|\mathcal{B}|}\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}\boldsymbol{\Sigma}_\mathcal{B}^{-1}\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}_\mathcal{B}}\boldsymbol{\sigma}_\mathcal{B}^{-2}\boldsymbol{\mu}_\mathcal{B}}_{\mathbf{m}_\mathcal{B}} \\
&\quad + \mathrm{tr}\left\{\boldsymbol{\Sigma}_\mathcal{B}^{-1}\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}\right\} + \log\frac{|\boldsymbol{\Sigma}_\mathcal{B}|}{|\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}|} - M\Bigg] \\
&= \frac{1}{2}\left[\tfrac{N^2}{|\mathcal{B}|^2}\bar{\mathbf{m}}_\mathcal{B}^\top \mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}\bar{\mathbf{m}}_\mathcal{B} + \mathrm{tr}\left\{\boldsymbol{\Sigma}_\mathcal{B}^{-1}\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}\right\} + \log\frac{|\boldsymbol{\Sigma}_\mathcal{B}|}{|\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}|} - M\right].
\end{aligned}$$

## B.2 Computational complexity

### B.2.1 General complexity

The proposed framework accommodates generic inducing features. For general inducing features, our method has computational complexity $\boxed{\mathcal{O}(M^3 + M^2|\mathcal{B}|)}$, matching the asymptotic scaling of GPVAE variants based on conventional inducing points. In practice, both $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}$ and $\boldsymbol{\Sigma}_\mathcal{B}$ require a Cholesky decomposition to obtain the terms above. An efficient way to compute the training objective $\mathcal{L}_{\text{IDGPVAE}}$ is to *reuse* the Cholesky factors and the relevant terms as much as possible. Specifically, we use four triangular solves in our method, denoted as `solve_tri`, to simplify the computation:

$$\begin{aligned}
\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}} &= \mathbf{L}_\mathbf{K}\mathbf{L}_\mathbf{K}^\top, \quad \boldsymbol{\Sigma}_\mathcal{B} = \mathbf{L}_\boldsymbol{\Sigma}\mathbf{L}_\boldsymbol{\Sigma}^\top, \\
m_s(\mathbf{x}_n) &= \tfrac{N}{|\mathcal{B}|}\underbrace{\mathbf{K}_{z_n\bar{\mathbf{z}}}\mathbf{L}_\boldsymbol{\Sigma}^{-\top}}_{\texttt{solve_tri_2}}\underbrace{\mathbf{L}_\boldsymbol{\Sigma}^{-1}\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}_\mathcal{B}}\boldsymbol{\sigma}_\mathcal{B}^{-2}\boldsymbol{\mu}_\mathcal{B}}_{\texttt{solve_tri_1}}, \\
\Sigma_s(\mathbf{x}_n) &= k_{z_nz_n} + \underbrace{\mathbf{K}_{z_n\bar{\mathbf{z}}}\mathbf{L}_\boldsymbol{\Sigma}^{-\top}}_{\texttt{solve_tri_2}}\underbrace{\mathbf{L}_\boldsymbol{\Sigma}^{-1}\mathbf{K}_{\bar{\mathbf{z}}z_n}}_{\texttt{solve_tri_2}} - \underbrace{\mathbf{K}_{z_n\bar{\mathbf{z}}}\mathbf{L}_\mathbf{K}^{-\top}}_{\texttt{solve_tri_3}}\underbrace{\mathbf{L}_\mathbf{K}^{-1}\mathbf{K}_{\bar{\mathbf{z}}z_n}}_{\texttt{solve_tri_3}}, \\
\bar{\mathbf{m}}_\mathcal{B}^\top\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}\bar{\mathbf{m}}_\mathcal{B} &= \bar{\mathbf{m}}_\mathcal{B}^\top\mathbf{L}_\mathbf{K}\mathbf{L}_\mathbf{K}^\top\bar{\mathbf{m}}_\mathcal{B}, \quad \mathbf{L}_\mathbf{K}^\top\bar{\mathbf{m}}_\mathcal{B} = \underbrace{\mathbf{L}_\mathbf{K}^\top\mathbf{L}_\boldsymbol{\Sigma}^{-\top}}_{\texttt{solve_tri_4}}\underbrace{\mathbf{L}_\boldsymbol{\Sigma}^{-1}\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}_\mathcal{B}}\boldsymbol{\sigma}_\mathcal{B}^{-2}\boldsymbol{\mu}_\mathcal{B}}_{\texttt{solve_tri_1}}, \\
\mathrm{tr}\left\{\boldsymbol{\Sigma}_\mathcal{B}^{-1}\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}\right\} &= \mathrm{tr}\{\underbrace{\mathbf{L}_\mathbf{K}^\top\mathbf{L}_\boldsymbol{\Sigma}^{-\top}}_{\texttt{solve_tri_4}}\underbrace{\mathbf{L}_\boldsymbol{\Sigma}^{-1}\mathbf{L}_\mathbf{K}}_{\texttt{solve_tri_4}}\}, \\
\log|\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}| &= 2\log|\mathbf{L}_\mathbf{K}|, \quad \log|\boldsymbol{\Sigma}_\mathcal{B}| = 2\log|\mathbf{L}_\boldsymbol{\Sigma}|,
\end{aligned} \tag{16}$$

where $\boldsymbol{\Sigma}_\mathcal{B}{=}\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}} + \frac{N}{|\mathcal{B}|}\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}_\mathcal{B}}\boldsymbol{\sigma}_\mathcal{B}^{-2}\mathbf{K}_{\mathbf{z}_\mathcal{B}\bar{\mathbf{z}}}$. Here, the covariance matrices $\mathbf{K}_{z_n\bar{\mathbf{z}}}$ and $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}$ no longer correspond to standard kernel evaluations; rather, they represent inter-domain covariances between latent variables and inducing features.

### B.2.2 Complexity with Matérn RKHS inducing features

If the inducing features are defined via a Matérn RKHS inner product with fixed basis functions, the cross-covariance matrix $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}$ contains no trainable parameters and can therefore be precomputed once before training to reduce computational cost. Moreover, the matrices $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}$ and $\boldsymbol{\Sigma}_\mathcal{B}$ may have additional computational structure, depending on the chosen basis functions. In the following, we detail the computational complexity of IDGPVAE-VFF and IDGPVAE-BSF used in the main text.

**Complexity with IDGPVAE-VFF** Assume the auxiliary input $\mathbf{X}$ has $D$ dimensions, and for each input dimension, we use $M$ inducing frequencies. This leads to $D(2M{+}1)$ inducing variables $\bar{\mathbf{z}} = [\bar{z}_{m,d}]_{m=1,d=1}^{2M+1,D}$. We use additive kernels when $\mathbf{X}$ is multi-dimensional.

The cross-covariance matrix $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}} \in \mathbb{R}^{D(2M+1)\times N}$ for the entire dataset can be precomputed. Sub-matrices $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}_\mathcal{B}}$ are columns extracted from $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}$ during mini-batch training. By construction, the

covariance between inducing variables $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}$ is a $D$-block-diagonal matrix with each block low-rank-plus-diagonal. This block-diagonal structure can be exploited in the Cholesky decomposition and the subsequent triangular solves, since the lower-triangular Cholesky factor $\mathbf{L}_{\mathbf{K}}$ is also block diagonal. Additionally, it is also beneficial to avoid Cholesky decomposition of $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}$ in `solve_tri_3` and instead compute $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}^{-1}\mathbf{K}_{\bar{\mathbf{z}}z_n}$ directly by the Woodbury matrix identity (as shown in Eq. 22), which costs $\mathcal{O}(DM|\mathcal{B}|)$. The matrix $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}$ contains kernel parameters, so it must be recomputed at every training iteration.

The matrix $\mathbf{\Sigma}_{\mathcal{B}}$ is usually dense, and its Cholesky decomposition requires $\mathcal{O}(D^3M^3)$ complexity for multi-dimensional inputs. Alternatively, this cubic bottleneck in $DM$ can be bypassed by applying the Woodbury matrix identity for the inverse $\mathbf{\Sigma}_{\mathcal{B}}^{-1}$, because $\mathbf{\Sigma}_{\mathcal{B}}$ is a low-rank update (of rank $|\mathcal{B}|$) to the strictly block-diagonal prior covariance $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}$. This shifts the dense inversion to an inner $|\mathcal{B}|\times|\mathcal{B}|$ matrix. When the mini-batch size is much smaller than the total number of inducing features ($|\mathcal{B}|\ll DM$), this reduces the complexity from $\mathcal{O}(D^3M^3)$ to $\mathcal{O}(|\mathcal{B}|^3+DM^2|\mathcal{B}|)$. In summary, the Cholesky-based strategy incurs a cubic cost in $DM$, whereas the Woodbury-based strategy incurs a cubic cost in the mini-batch size $|\mathcal{B}|$.

**Complexity with IDGPVAE-BSF** We employ $M$ B-spline basis functions $B_{m,p}(\cdot)$ of degree $p$ with uniform knots for IDGPVAE-BSF. We choose $[a,b]$ such that $[v_p, v_M] \subseteq [a,b]$. We analyse the computational complexity separately for one-dimensional and multi-dimensional auxiliary inputs.

- *One-dimensional* $\mathbf{X} \in \mathbb{R}^{N\times 1}$

In this case, the compact support of B-spline bases gives a sparse cross-covariance $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}$, with at most $p+1$ non-zero entries per column. Consequently, the mini-batch update $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}_{\mathcal{B}}}\boldsymbol{\sigma}_{\mathcal{B}}^{-2}\mathbf{K}_{\mathbf{z}_{\mathcal{B}}\bar{\mathbf{z}}}$ is banded. Together with the band matrix $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}$, the matrix $\mathbf{\Sigma}_{\mathcal{B}}$ therefore retains a banded structure with bandwidth $p$. The cross-covariance $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}$ can be computed only once before training, and, using a compact storage format, its memory footprint is only $\mathcal{O}(N(p+1))$. The mini-batch version $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}_{\mathcal{B}}}$ is obtained by selecting relevant columns of $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}$, each of which contains $p+1$ consecutive non-zero entries.

Exploiting the banded structure of $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}$ and $\mathbf{\Sigma}_{\mathcal{B}}$, their Cholesky decompositions only cost $\mathcal{O}(Mp^2)$ and require $\mathcal{O}(M(p+1))$ storage (as illustrated in Figure 5). Since the Cholesky factors $\mathbf{L}_{\mathbf{K}}$ and $\mathbf{L}_{\mathbf{\Sigma}}$ are both lower triangular with bandwidth $p$ (stored in $\mathcal{O}(M(p+1))$), the related triangular-solving operations and matrix multiplications will benefit from that. Specifically, the computational complexity of `solve_tri_1~3` is $\mathcal{O}(M|\mathcal{B}|p)$ for a general dense right-hand side. The trace term in (16) is computed through `solve_tri_4`. Although both Cholesky factors are banded, this triangular solve generally produces a dense lower-triangular matrix. Its direct evaluation costs $\mathcal{O}(M^2p)$ time, and $\mathcal{O}(M^2)$ storage if the result is materialised. Thus, this step scales quadratically in $M$. Even so, it remains a substantial improvement over dense inducing-variable computations, which scale cubically with $M$, allowing much larger inducing resolutions in practice.

- *Multi-dimensional* $\mathbf{X} \in \mathbb{R}^{N\times D}$

When the inputs have $D$ dimensions, we introduce inducing variables at each input dimension, i.e., $\bar{\mathbf{z}} = [\bar{z}_{m,d}]_{m=1,d=1}^{M,D}$, and use an additive kernel for each output dimension in the latent space. The matrix $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}} \in \mathbb{R}^{DM\times DM}$ is band-diagonal with the same bandwidth as the one-dimensional case. Its Cholesky decomposition costs $\mathcal{O}(DMp^2)$, and the relevant matrix multiplication can be further simplified. However, since the non-zero entries in each column of $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}$ do not appear consecutively, the term $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}_{\mathcal{B}}}\boldsymbol{\sigma}_{\mathcal{B}}^{-2}\mathbf{K}_{\mathbf{z}_{\mathcal{B}}\bar{\mathbf{z}}}$ is no longer banded. As a result, the matrix $\mathbf{\Sigma}_{\mathcal{B}}$ is not banded. Similarly, the dominant cost can be cubic in $DM$ (via the Cholesky-based approach) or in $|\mathcal{B}|$ (via the Woodbury-based approach).

Table 6 summarises the dominant computational complexity. For one-dimensional inputs, IDGP-VAE–VFF preserves the cubic dependence on $M$ typical of SVGPVAE, whereas IDGPVAE-BSF replaces the dense cubic bottleneck with banded factorisations and a structured quadratic trace term in the current implementation. For multi-dimensional inputs, the leading computational cost for both models is $\mathcal{O}(D^3M^3)$ or $\mathcal{O}(|\mathcal{B}|^3)$. This bottleneck could be alleviated by choosing alternative inducing features, which we leave for future work.

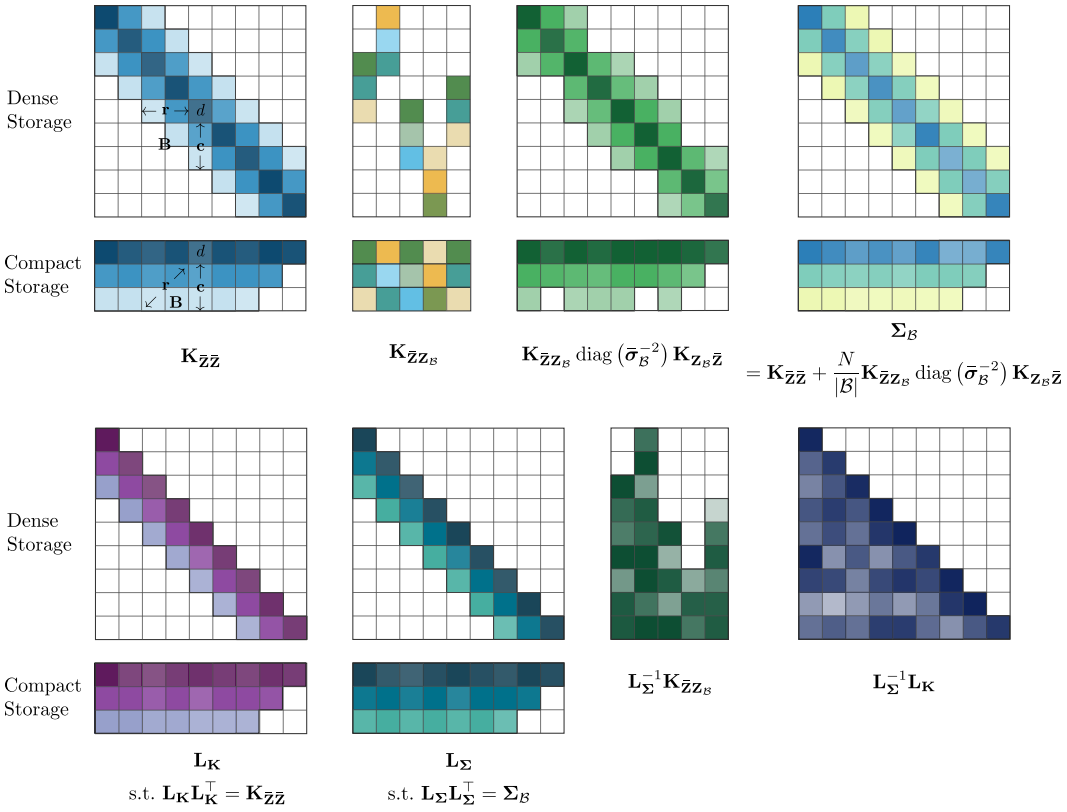



**Figure 5:** Key matrix structures in IDGPVAE-BSF and their compact band storage layouts. **(Left):** Non-zero entries of the banded matrices (e.g., $\mathbf{K}_{\bar{\mathbf{Z}}\bar{\mathbf{Z}}}, \mathbf{L}_\mathbf{K} \in \mathbb{R}^{M\times M}$) are mapped diagonal-wise into a $(p+1)\times M$ array to optimise memory footprint. **(Top-left):** The `level2partition` function extracts four components $(d, \mathbf{r}, \mathbf{c}, \mathbf{B})$ from the matrix to facilitate banded Cholesky decomposition. **(Top-right):** The sparse structure of $\mathbf{K}_{\bar{\mathbf{Z}}\mathbf{Z}_\mathcal{B}}$, characterised by at most $p+1$ consecutive non-zero entries per column, yields a banded covariance matrix $\boldsymbol{\Sigma}_\mathcal{B}$ in IDGPVAE-BSF. **(Bottom-right):** The result of the triangular solve $\mathbf{L}_\boldsymbol{\Sigma}^{-1}\mathbf{K}_{\bar{\mathbf{Z}}\mathbf{Z}_\mathcal{B}}$ shows a sawtooth pattern induced by $\mathbf{K}_{\bar{\mathbf{Z}}\mathbf{Z}_\mathcal{B}}$. Solving a banded system with a banded right-hand side (i.e., $\mathbf{L}_\boldsymbol{\Sigma}^{-1}\mathbf{L}_\mathbf{K}$) results in a dense lower triangular matrix.

**Table 6:** Computational complexity of IDGPVAE, compared with the standard inducing-point method.

| MODEL | $D=1$ | $D>1$ |
|---|---|---|
| SVGPVAE | $\mathcal{O}(M^3+M^2\lvert\mathcal{B}\rvert)$ | |
| General IDGPVAE | $\mathcal{O}(M^3+M^2\lvert\mathcal{B}\rvert)$ | |
| IDGPVAE-VFF | $\mathcal{O}(M^3+M^2\lvert\mathcal{B}\rvert)$ | $\mathcal{O}(D^3M^3)$ or $\mathcal{O}(\lvert\mathcal{B}\rvert^3)$ |
| IDGPVAE-BSF | $\mathcal{O}(M^2p)$ | $\mathcal{O}(D^3M^3)$ or $\mathcal{O}(\lvert\mathcal{B}\rvert^3)$ |

# C Inducing features

This section details the inter-domain inducing features used in IDGPVAE and their properties, focusing on Matérn RKHS inducing features. We first review Matérn kernels and their associated RKHS inner products (Appendix C.1), then describe covariance structures for two instantiations, i.e., VFF (Appendix C.2) and BSF (Appendix C.3).

### C.1 Matérn family and RKHS inner product

We take the Matérn-$\frac{3}{2}$ kernel as a representative example:

$$k_{3/2}(r) = \sigma^2(1+\lambda r)\exp(-\lambda r), \quad \lambda = \frac{\sqrt{3}}{l},$$

where $r = |x - x'|$, $\sigma^2$ is the kernel variance, and $l$ is the lengthscale. Durrande et al. (2016) derive closed-form expressions for the RKHS inner products associated with the Matérn–$\frac{1}{2}$, $\frac{3}{2}$, and $\frac{5}{2}$ kernels on a real interval $[a, b]$. For completeness, we write here the expression for the Matérn–$\frac{3}{2}$ case:

$$\langle f, \phi\rangle_{\mathcal{H}_{\frac{3}{2}}} = \frac{1}{4\lambda^3\sigma^2}\int_a^b (\lambda\mathcal{I}+\mathcal{D})^2(f)(\lambda\mathcal{I}+\mathcal{D})^2(\phi)\,\mathrm{d}x + \frac{1}{\sigma^2}f(a)\phi(a) + \frac{1}{\lambda^2\sigma^2}f'(a)\phi'(a), \quad (17)$$

where $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ represents the inner product of RKHS $\mathcal{H}$, $\mathcal{I}(\phi) := \phi$ is the identity operator, $\mathcal{D}(\phi) := \phi'$ is the differentiation operator, and $(\lambda\mathcal{I}+\mathcal{D})^2(\phi)$ stands for $\lambda^2\phi + 2\lambda\phi' + \phi''$. The reproducing property of the Matérn inner products can be verified by

$$\langle k(x,\cdot), \phi(\cdot)\rangle_{\mathcal{H}} = \phi(x), \quad x \in [a,b],\ \phi \in \mathcal{H}.$$

The expression (17) provides alternative transformations for building inter-domain GPs (Hensman et al., 2018; Dutordoir et al., 2020; Cunningham et al., 2023). Specifically, the inducing variables $\bar{z}_m$ can be obtained through projections of the GP $z$ onto a set of basis functions $\{\phi_m(\cdot)\}_{m=1}^M$:

$$\bar{z}_m = \mathcal{P}_m[z] = \langle z, \phi_m\rangle_{\mathcal{H}}. \quad (18)$$

In other words, the linear functional $\mathcal{P}_m$ in the main text is now expressed explicitly via the Matérn RKHS inner product.

The RKHS inner product in (17) is of particular interest in inter-domain GPs, as it enables us to compute the covariances between the inter-domain variables $\bar{z}$ and the GP $z$ analytically. Taking advantage of the linearity of the transformation and the reproducing property, the covariance of the function value and the inter-domain variable corresponds precisely to the element of the basis functions:

$$\mathrm{cov}\big(\bar{z}_m, z(x)\big) = \mathrm{cov}\big(\langle z(\cdot), \phi_m(\cdot)\rangle_{\mathcal{H}}, z(x)\big) = \langle k(x,\cdot), \phi_m(\cdot)\rangle_{\mathcal{H}} = \phi_m(x), \quad (19)$$

which is valid for $x \in [a, b]$. Similarly, the covariance of the inter-domain variables is given by

$$\mathrm{cov}(\bar{z}_m, \bar{z}_{m'}) = \mathrm{cov}\big(\langle z, \phi_m\rangle_{\mathcal{H}}, \langle z, \phi_{m'}\rangle_{\mathcal{H}}\big) = \langle \phi_m, \phi_{m'}\rangle_{\mathcal{H}}, \quad (20)$$

which can be computed analytically using the inner-product expression. For simplicity, we restrict the inputs to $x \in [a, b]$ and discuss a single latent channel of IDGPVAE in what follows.

### C.2 Variational Fourier feature

**Cross-covariance $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}$** VFF uses the basis functions $\{\phi_m\}_{m=0}^{2M}$ as a truncated Fourier basis (Hensman et al., 2018):

$$\phi_m(x) = \begin{cases} 1, & m = 0, \\ \cos(\omega_m(x-a)), & m = 1, ..., M, \\ \sin(\omega_{m-M}(x-a)), & m = M+1, ..., 2M, \end{cases} \qquad x \in [a,b], \quad (21)$$

where $\omega_m = \frac{2\pi m}{b-a}$. $\{\omega_m\}_{m=1}^M$ are known as $M$ harmonic *inducing frequencies*, analogous to inducing locations of SVGP. We collect the cross-covariance entries $\mathrm{cov}\big(\bar{z}_m, z(x)\big)$ into a matrix denoted as $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}} \in \mathbb{R}^{(2M+1)\times N}$, where $N$ is the number of data points and the $(m,n)$-th entry is given by $[\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}]_{m,n} = \phi_m(x_n)$.

*Remark* C.1 (Cached $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}$). If the inputs $\mathbf{X}$ are constrained to the interval $[a, b]$, the matrix $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}$ (or $\mathbf{K}_{\mathbf{z}\bar{\mathbf{z}}}$) can be computed only once before optimisation (i.e., precomputation) since it does not depend on trainable parameters. This precomputation reduces per-epoch training cost, since only the kernel hyperparameters in $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}$ need to be updated.

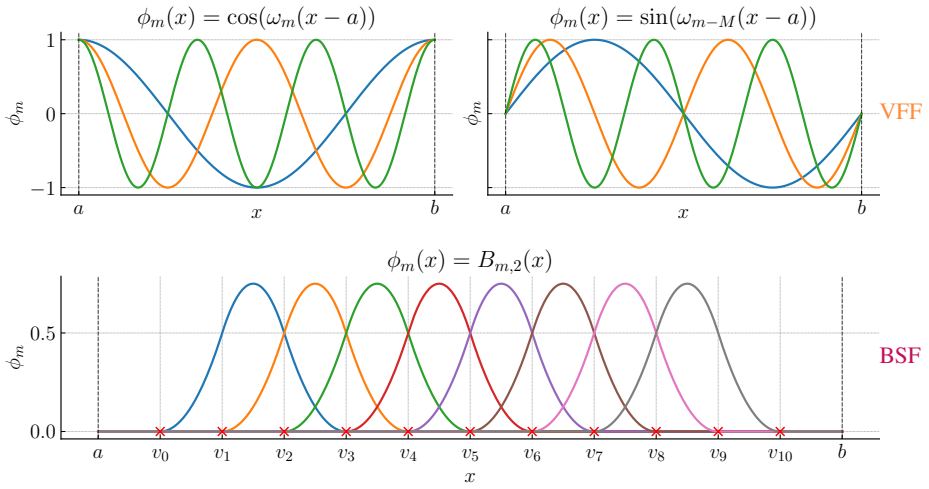



**Figure 6:** Illustration of the basis functions used in our inter-domain constructions: VFF (top) and BSF (bottom). VFF employs global sinusoidal bases, while BSF uses compactly supported B-splines, leading to structured inter-domain covariance (see also Figure 5). Red crosses represent the knots of BSFs. Throughout this work, $[a, b]$ is chosen to cover the data range and the supports of all BSFs to mitigate edge effects.

**Covariance $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}$** The $(m, m')$-th entry of the covariance matrix $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}} \in \mathbb{R}^{(2M+1)\times(2M+1)}$ can be derived in closed form by substituting $\phi_m$ and $\phi_{m'}$ in (20) appropriately. The covariance $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}$ can be written as a block-diagonal matrix whose sub-matrix has a low-rank-plus-diagonal form:

$$\begin{aligned}
\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}} &= \text{blockdiag}\left(\mathbf{K}^{\cos}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}, \mathbf{K}^{\sin}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}\right), \\
\mathbf{K}^{\cos}_{\bar{\mathbf{z}}\bar{\mathbf{z}}} &= \text{diag}(\boldsymbol{\alpha}_{0:M}) + \boldsymbol{\beta}_{0:M}\boldsymbol{\beta}^\top_{0:M} \in \mathbb{R}^{(M+1)\times(M+1)}, \\
\mathbf{K}^{\sin}_{\bar{\mathbf{z}}\bar{\mathbf{z}}} &= \text{diag}(\boldsymbol{\alpha}_{M+1:2M}) + \boldsymbol{\beta}_{M+1:2M}\boldsymbol{\beta}^\top_{M+1:2M} \in \mathbb{R}^{M\times M},
\end{aligned}$$

where $\boldsymbol{\alpha}_{0:M} = [\alpha_0, \ldots, \alpha_M]$ and so on. Specifically, we have the following for the Matérn-$\frac{3}{2}$ kernel,

$$\begin{cases}
\alpha_0 = \dfrac{b-a}{s_{\frac{3}{2}}(0)}, & m = 0 \\
\alpha_m = \dfrac{b-a}{2s_{\frac{3}{2}}(\omega_m)} & m = 1, \cdots, M \\
\alpha_m = \dfrac{b-a}{2s_{\frac{3}{2}}(\omega_{m-M})}, & m = M+1, \cdots, 2M
\end{cases}
\qquad
\begin{cases}
\beta_m = \dfrac{1}{\sigma} & m = 0, 1, \cdots, M, \\
\beta_m = \dfrac{\omega_{m-M}}{\sigma\lambda} & m = M+1, \cdots, 2M,
\end{cases}$$

where $s_{\frac{3}{2}}$ is essentially the spectral density of the kernel (Rahimi and Recht, 2007): $s_{\frac{3}{2}}(\omega) = \frac{4\sigma^2\lambda^3}{(\lambda^2+\omega^2)^2}$.

Due to the special covariance structure in $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}$, solving $\mathbf{K}^{-1}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}$ can be computed block-wise in $\mathcal{O}(MN)$ operations instead of the standard $\mathcal{O}(M^2N)$ by the Woodbury matrix identity. Take the block $\mathbf{K}^{\sin}_{\bar{\mathbf{z}}\mathbf{z}}$ as an example

$$\mathbf{K}^{-1}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}} = \underbrace{\mathbf{D}^{-1}\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}}_{\mathcal{O}(MN)} - \underbrace{\mathbf{D}^{-1}\boldsymbol{\beta}}_{\mathcal{O}(Mr)}\left(\mathbf{I} + \boldsymbol{\beta}^\top\mathbf{D}^{-1}\boldsymbol{\beta}\right)^{-1}\underbrace{\boldsymbol{\beta}^\top\mathbf{D}^{-1}\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}}_{\mathcal{O}(rMN)}, \tag{22}$$

where we suppress the block superscript, and $\mathbf{D}$ and $\boldsymbol{\beta}$ represent the diagonal and the low-rank matrices, respectively. $r$ is the rank of $\boldsymbol{\beta} \in \mathbb{R}^{M\times r}$. Then, the Cholesky decomposition is applied to the smaller intermediate matrix.

**Higher-dimensional inputs** While the above derivation focuses on Matérn kernels on a scalar input space, the method extends to higher-dimensional inputs by summing kernels. Specifically,

using a sum of independent GPs on each input $\mathbf{x} = [x_1, \ldots, x_D] \in \mathbb{R}^D$ leads to an additive kernel (Duvenaud et al., 2011):

$$z(\mathbf{x}) = \sum_{d=1}^{D} z_d(x_d), \quad z_d \sim \mathcal{GP}(0, k_d(x_d, x_d')) \Longrightarrow z(\mathbf{x}) \sim \mathcal{GP}\left(0, \sum_{d=1}^{D} k_d(x_d, x_d')\right),$$

where each $k_d$ is a one-dimensional Matérn kernel and the latent processes $z_d$ are assumed independent across input dimensions. Assuming each input dimension has $M$ frequencies, we will have a cross-covariance matrix $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}} \in \mathbb{R}^{(2M+1)D \times N}$ from $(2M+1)D$ inducing variables $\bar{z}_{m,d}$, whose element is given by

$$\operatorname{cov}\big(\bar{z}_{m,d}, z(\mathbf{x})\big) = \operatorname{cov}\left(\bar{z}_{m,d}, \sum_{d=1}^{D} z_d(x_d)\right) \overset{\text{ind.}z_d}{=} \operatorname{cov}\left(\bar{z}_{m,d}, z_d(x_d)\right),$$

since the inter-domain variable $\bar{z}_{m,d}$ is independent of $z_{d' \neq d}$. Furthermore, $\bar{z}_{m,d}$ from different sub-GPs are independent by construction, so $\operatorname{cov}\big(\bar{z}_{m,d}, \bar{z}_{m',d'}\big) = 0$ if $d \neq d'$.

### C.3 B-spline feature

Let $V = \{v_m\}_{m=0}^{M+p}$ be a set of non-decreasing "knots" such that $v_0 \leqslant v_1 \leqslant \cdots \leqslant v_{M+p}$. A knot vector is the sequence of knot values given by $[v_0, v_1, \cdots, v_{M+p}]$. A B-spline of order $p+1$ is a collection of piecewise polynomial functions of degree $p$, denoted as $\{B_{m,p}(\cdot)\}_{m=0}^{M-1}$. In particular, B-spline basis functions can be defined recursively by the Cox-de Boor recursion formula (de Boor, 1978; Prautzsch et al., 2002). For clarity and simplicity, this paper focuses on *uniform* or *cardinal* B-splines, meaning all knot spans $[v_m, v_{m+1})$ are equal (see Figure 6, bottom).

*Remark* C.2. By induction, the $m$-th B-spline basis function $B_{m,p}$ is zero outside the interval $(v_m, v_{m+p+1})$. This implies that the closed support of $B_{m,p}$ is $[v_m, v_{m+p+1}]$, which includes $p+2$ knots. There are $M$ basis functions whose supports collectively cover the range $[v_0, v_{M+p}]$. Each B-spline basis function overlaps with at most $2p$ other basis functions, since 2 basis functions protrude from every knot inside $(v_m, v_{m+p+1})$. Most importantly, given a fixed $x \in [v_m, v_{m+1})$, there are at most $p+1$ non-zero B-spline function values:

$$B_{m-p,p}(x), B_{m-p+1,p}(x), \ldots, B_{m,p}(x).$$

*Remark* C.3. The B-spline $B_{m,p}(x)$ has $(p-1)$-th continuous derivatives for $p \geqslant 2$; that is, $B_{m,p}(x)$ is of differentiability class $C^{p-1}$. Moreover, $B_{m,p}(x)$ is $p$-times weakly differentiable for $p \geqslant 1$.

**Cross-covariance $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}$** The BSF inducing features are introduced by considering $\{\phi_m\}_{m=0}^{M-1}$ to be B-spline basis functions of degree $p$: $\phi_m(\cdot) = B_{m,p}(\cdot)$ (Cunningham et al., 2023). Therefore, the associated entry of $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}$ is given by $[\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}]_{m,n} = \phi_m(x_n) = B_{m,p}(x_n)$ for $x_n \in [a, b]$. Based on this construction and the characteristics of B-spline functions, we can draw the following conclusions.

*Property* C.1 (Sparse $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}$). $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}$ is a sparse matrix. Each column of $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}$ has at most $p+1$ non-zero elements at consecutive matrix locations. If $x_n \in [v_m, v_{m+1})$, the (zero-based) row indices of the corresponding non-zero entries of the $n^{\text{th}}$ column $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}$ are $\{m-p, \ldots, m-p+1, \ldots, m\}$. This follows directly from the compact support of the B-spline basis functions (Remark C.2).

*Property* C.2 (Banded $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}\mathbf{K}_{\mathbf{z}\bar{\mathbf{z}}}$). Built on the Matérn projection onto B-splines of degree $p$, the matrix product $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}\mathbf{K}_{\mathbf{z}\bar{\mathbf{z}}}$ is a banded matrix with bandwidth at most $p$. This property can be proven by the outer-product expansion of $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}\mathbf{K}_{\mathbf{z}\bar{\mathbf{z}}}$.

*Remark* C.4 (Cached $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}$). Similar to Remark C.1, but we can cache it in a more compact storage format.

**Covariance $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}$** The covariance between B-spline inducing variables is given by $[\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}]_{m,m'} = \langle \phi_m(\cdot), \phi_{m'}(\cdot) \rangle_{\mathcal{H}} = \langle B_{m,p}(\cdot), B_{m',p}(\cdot) \rangle_{\mathcal{H}}$. The covariance matrix $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}$ has the following property:

*Property* C.3 (Banded $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}$ and $\mathbf{\Sigma}_{\mathcal{B}}$). Under the Matérn projection defined by (17), the $(i,j)$-th element $[\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}]_{i,j}$ is zero if $B_{i,p}(\cdot)$, $B_{j,p}(\cdot)$ do not have any intersecting support. Therefore, the covariance $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}$ is a banded matrix with bandwidth at most $p$. Combining Property C.2, the matrix $\mathbf{\Sigma}_{\mathcal{B}}$ is also a banded matrix with bandwidth $p$.

The locality of B-splines induces banded structures in the covariance matrices of IDGPVAE-BSF. The banded structures can be exploited for both storage and computational efficiency. Figure 5 visualises the matrices discussed above in the context of IDGPVAE. For details of the operators for banded matrices, see Appendix D.

# D Custom differentiable banded operators

This section describes the C++/CUDA custom banded linear algebra operators used in this work. The implementation includes a compact storage format, forward computation, and reverse-mode differentiation.

## D.1 Motivation and design requirements

The BSF instantiation produces symmetric positive definite banded matrices with a small bandwidth determined by the spline degree. Although these matrices can be stored as dense tensors, doing so discards the structure that underlies BSF's computational advantage. Most existing banded-matrix libraries also do not meet our needs: the package of Durrande et al. (2019) targets legacy TensorFlow, SciPy does not support automatic differentiation (Virtanen et al., 2020), and although the NUMML package (Nytko et al., 2025) provides autograd support in PyTorch, it lacks some operations required by our framework, such as Cholesky decomposition. We therefore implement a small set of specialised banded primitives for the compact storage format used in IDGPVAE-BSF. As illustrated in Figure 5, we adopt the banded matrix storage format introduced by Durrande et al. (2019).

While the custom operators used here are model-agnostic, they are tailored to compact banded matrices and differentiable solve patterns arising from the BSF covariance structure, rather than intended as a general-purpose sparse linear algebra package. We summarise our model requirements for banded operators in Table 7.

**Table 7:** A summary of Banded-operator requirements in IDGPVAE-BSF.

| Requirement | Role in IDGPVAE-BSF | Implemented support |
|---|---|---|
| Compact banded storage | Stores matrices with fixed bandwidth without materialising dense $M \times M$ tensors. | Batched lower-banded layout (Appendix D.2). |
| Banded Cholesky decomposition | Factorises banded $\mathbf{K}_{\bar{\mathbf{z}}\bar{\mathbf{z}}}$ and $\mathbf{\Sigma}_{\mathcal{B}}$ used in posterior and KL computations. | C++ forward operator utilising LAPACKE; tailored CUDA forward operator (Appendix D.4). |
| Banded triangular solves | Computes repeated solves involving Cholesky factors and structured right-hand sides. | C++/CUDA forward operators built on the compact storage (Appendix D.5). |
| Reverse-mode differentiation | Enables end-to-end training of all parameters without dense fallbacks for dominant operations. | Custom backward rules integrated with PyTorch autograd on CPU/GPU (Appendix D.3). |
| GPU execution | Supports large inducing scales on the GPU alongside other neural components for joint training. | Custom NVIDIA CUDA kernels for core operations (Appendix D.6). |

## D.2 Compact banded storage

**Banded matrix** A banded matrix is a sparse matrix whose non-zero entries are confined to a diagonal band around the main diagonal. The *bandwidth* $p$ of a banded matrix denotes the number of non-zero subdiagonals below the main diagonal (the upper triangle follows by symmetry). Since the variational posterior of IDGPVAE-BSF is associated with positive-definite (p.d.) banded matrices, we exploit their sparse structure for storage and computational gains. In this section, we focus on symmetric p.d. banded matrices and their corresponding lower triangular banded Cholesky factors.

**Storage format** In this work, instead of using a general sparse storage format such as CSR, we adopt a diagonal-wise compact storage format for batched banded matrices, following Durrande et al. (2019). As illustrated in Figure 5, a banded matrix $\mathbf{A} \in \mathbb{R}^{M \times M}$[3] with bandwidth $p$ is stored in a $(p+1) \times M$ array $\mathbf{A}^b$, where each column corresponds to a column of $\mathbf{A}$ and each row corresponds to a diagonal. For 1-based indices and $i \geqslant j$, entries below the main diagonal satisfy

$$A_{ij} = A^b_{i-j+1,j}, \quad \text{or equivalently} \quad A_{i+j-1,j} = A^b_{ij}.$$

To preserve algorithmic clarity, we describe all matrix operations in the following subsections using canonical dense indexing rather than compact storage offsets.

### D.3 Differentiation of matrix operations

Before discussing the banded-matrix operations used in our model in detail, we first briefly explain the backpropagation mechanism for generic matrix operators. Consider a chain of computations that begins with two leaf matrices $\mathbf{A}$ and $\mathbf{B}$, followed by an intermediate matrix $\mathbf{C}$, and ends with a scalar output $y \in \mathbb{R}$:

$$\mathbf{A}, \mathbf{B} \xrightarrow{\boldsymbol{f}} \mathbf{C} \longrightarrow y.$$

The first computation step $\boldsymbol{f}$ and its differential can be expressed by

$$\mathbf{C} = \boldsymbol{f}(\mathbf{A}, \mathbf{B}) \Longrightarrow \mathrm{d}\mathbf{C} = \frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\mathbf{A}}\,\mathrm{d}\mathbf{A} + \frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\mathbf{B}}\,\mathrm{d}\mathbf{B},$$

where $[\mathrm{d}\mathbf{A}]_{ij} = \mathrm{d}A_{ij}$ and $[\mathrm{d}\mathbf{B}]_{ij} = \mathrm{d}B_{ij}$. The derivatives $\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\mathbf{A}}$ and $\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\mathbf{B}}$ can be derived by differentiation formulas. This reflects how the variable of interest would change under an infinitesimal perturbation of the leaf nodes. The commonly used gradient backpropagation utilises the "reverse mode" (Giles, 2008), which starts at the end of the computational graph and summarises the effect of perturbing the final output $y$ in terms of matrices associated with intermediate variables. Assuming $\mathbf{A}$ and $\mathbf{B}$ are leaf nodes not used in other intermediate computations, we have

$$\mathrm{d}y = \sum_{i,j} \bar{C}_{ij}\,\mathrm{d}C_{ij} = \mathrm{tr}\left(\bar{\mathbf{C}}^\top \mathrm{d}\mathbf{C}\right) = \mathrm{tr}\left(\bar{\mathbf{C}}^\top \frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\mathbf{A}}\,\mathrm{d}\mathbf{A}\right) + \mathrm{tr}\left(\bar{\mathbf{C}}^\top \frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\mathbf{B}}\,\mathrm{d}\mathbf{B}\right),$$

where we denote reverse-mode sensitivities by $\bar{\mathbf{C}}$, with entries $\bar{C}_{ij} = \frac{\partial y}{\partial C_{ij}}$, following the notation of Giles (2008), Murray (2016), and Durrande et al. (2019). Since $\mathrm{d}y = \mathrm{tr}\left(\bar{\mathbf{A}}^\top \mathrm{d}\mathbf{A}\right) + \mathrm{tr}\left(\bar{\mathbf{B}}^\top \mathrm{d}\mathbf{B}\right)$, we can read off the coefficients

$$\bar{\mathbf{A}} = \frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\mathbf{A}^\top}\bar{\mathbf{C}}, \quad \bar{\mathbf{B}} = \frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\mathbf{B}^\top}\bar{\mathbf{C}}.$$

### D.4 Banded Cholesky decomposition

This subsection introduces the autograd operator for Cholesky decomposition of banded matrices.

**Forward pass** This routine can be performed *in-place*, as shown in Algorithm 1. Specifically, it iterates over the columns of the input matrix $\mathbf{A}$, replacing the lower-triangular part with its Cholesky decomposition. In each iteration, the routine `level2partition` returns four views $(\mathbf{r}, d, \mathbf{B}, \mathbf{c})$ onto the current banded matrix, corresponding to the elements required at step $i$ of the Cholesky–Banachiewicz algorithm (see Figure 5, top left). This allows an in-place factorisation of $\mathbf{A}$ into its lower-triangular Cholesky factor $\mathbf{L}$.

**Backward pass** Algorithmic update rules for the gradient backpropagation in Cholesky decomposition can be derived by analysing the differentiation of each loop in Algorithm 1:

$$\begin{aligned}
&\mathbf{L_c} = (\mathbf{A_c} - \mathbf{L_B}\mathbf{L_r}^\top)/L_d \\
&\Rightarrow \mathrm{d}\mathbf{L_c} = \frac{\mathrm{d}\mathbf{A_c} - \mathrm{d}\mathbf{L_B}\mathbf{L_r}^\top - \mathbf{L_B}\,\mathrm{d}\mathbf{L_r}^\top - \mathbf{L_c}\,\mathrm{d}L_d}{L_d} \\
&\Rightarrow \mathrm{tr}\left(\bar{\mathbf{L}}_\mathbf{c}^\top \mathrm{d}\mathbf{L_c}\right) = \mathrm{tr}\left(\frac{\bar{\mathbf{L}}_\mathbf{c}^\top}{L_d}\,\mathrm{d}\mathbf{A_c}\right) - \mathrm{tr}\left(\frac{(\bar{\mathbf{L}}_\mathbf{c}\mathbf{L_r})^\top}{L_d}\,\mathrm{d}\mathbf{L_B}\right) \\
&\qquad\qquad - \mathrm{tr}\left(\frac{(\bar{\mathbf{L}}_\mathbf{c}^\top\mathbf{L_B})^\top}{L_d}\,\mathrm{d}\mathbf{L_r}\right) - \mathrm{tr}\left(\frac{(\mathbf{L}_\mathbf{c}^\top\bar{\mathbf{L}}_\mathbf{c})^\top}{L_d}\,\mathrm{d}L_d\right);
\end{aligned} \tag{23}$$

[3] While our implementation supports batch dimensions, we do not illustrate the batch shape here for simplicity.

**Algorithm 1:** The in-place Cholesky–Banachiewicz algorithm for banded matrices

**Input:** An $M \times M$ symmetric banded matrix $\mathbf{A}$ with bandwidth $p$
**Output:** An $M \times M$ banded lower-triangular matrix $\mathbf{L}$ with bandwidth $p$

Initialise $\mathbf{L} = \texttt{tril}\,(\mathbf{A})$;
**for** $i = 1$ **to** $M$ **do**
  $\mathbf{r},\ d,\ \mathbf{B},\ \mathbf{c} = \texttt{level2partition}(\mathbf{L},\ i)$;
  $d \leftarrow \sqrt{d - \mathbf{r}\mathbf{r}^\top}$; `// diag entry`
  $\mathbf{c} \leftarrow (\mathbf{c} - \mathbf{B}\mathbf{r}^\top)/d$; `// non-diag column`
**return** $\mathbf{L}$;

$$
\begin{aligned}
& L_d = \sqrt{A_d - \mathbf{L_r}\mathbf{L_r}^\top} \\
& \Rightarrow \mathrm{d}L_d = \frac{\mathrm{d}A_d/2 - \mathrm{d}\mathbf{L_r}\mathbf{L_r}^\top}{L_d} && (24) \\
& \Rightarrow \mathrm{tr}\left(\bar{L}_d^\top\, \mathrm{d}L_d\right) = \mathrm{tr}\left(\frac{\bar{L}_d}{2L_d}\,\mathrm{d}A_d\right) - \mathrm{tr}\left(\frac{(\bar{L}_d\mathbf{L_r})^\top}{L_d}\,\mathrm{d}\mathbf{L_r}\right), && (25)
\end{aligned}
$$

where the subscript indicates one of the four positions from `level2partition`. Since the final output perturbation $\mathrm{d}y$ is accounted for by the contributions from the diagonal and strictly lower parts, i.e., $\mathrm{d}y = \sum \mathrm{tr}\left(\bar{\mathbf{L}}_\mathbf{c}^\top\, \mathrm{d}\mathbf{L_c}\right) + \sum \mathrm{tr}\left(\bar{L}_d\, \mathrm{d}L_d\right)$, the effect of each entry of $\mathrm{d}\mathbf{A}$ transmitted via $[\mathrm{d}L_d, \mathrm{d}\mathbf{L_c}]$ is *accumulated* when we reverse the loop order and the in-loop updates. Specifically, combining (23–25) and rearranging the coefficients (i.e., sensitivities) give

$$
\begin{aligned}
\mathrm{tr}\left(\bar{\mathbf{L}}_\mathbf{c}^\top\, \mathrm{d}\mathbf{L_c}\right) + \mathrm{tr}\left(\bar{L}_d\, \mathrm{d}L_d\right) = {} & \mathrm{tr}\left(\frac{1}{2L_d}\left[\bar{L}_d - \frac{\mathbf{L}_\mathbf{c}^\top \bar{\mathbf{L}}_\mathbf{c}}{L_d}\right]\mathrm{d}A_d\right) + \mathrm{tr}\left(\frac{\bar{\mathbf{L}}_\mathbf{c}^\top}{L_d}\,\mathrm{d}\mathbf{A_c}\right) \\
& - \mathrm{tr}\left(\left[\frac{1}{L_d}\left(\bar{L}_d - \frac{\mathbf{L}_\mathbf{c}^\top \bar{\mathbf{L}}_\mathbf{c}}{L_d}\right)\mathbf{L_r} + \frac{\bar{\mathbf{L}}_\mathbf{c}^\top \mathbf{L_B}}{L_d}\right]^\top \mathrm{d}\mathbf{L_r}\right) \\
& - \mathrm{tr}\left(\frac{(\bar{\mathbf{L}}_\mathbf{c}\mathbf{L_r})^\top}{L_d}\,\mathrm{d}\mathbf{L_B}\right).
\end{aligned}
$$

Based on the above equation, Algorithm 2 tracks and updates all the reverse-mode sensitivities in place (Murray, 2016). In implementation, only entries within the bandwidth $p$ of $\bar{\mathbf{A}}$ are updated.

**Algorithm 2:** Reverse-mode differentiation of Cholesky decomposition

**Input:** The Cholesky factor $\mathbf{L}$ of an $M \times M$ symmetric p.d. matrix $\mathbf{A}$, the reverse-mode sensitivity $\bar{\mathbf{L}}$
**Output:** The reverse-mode sensitivity $\bar{\mathbf{A}}$

Initialise $\bar{\mathbf{A}} = \bar{\mathbf{L}}$;
**for** $i \leftarrow M$ **downto** $1$ **do**
  $\mathbf{r},\ d,\ \mathbf{B},\ \mathbf{c} = \texttt{level2partition}(\mathbf{L},\ i)$;
  $\bar{\mathbf{r}},\ \bar{d},\ \bar{\mathbf{B}},\ \bar{\mathbf{c}} = \texttt{level2partition}(\bar{\mathbf{A}},\ i)$;
  $\bar{\mathbf{c}} \leftarrow \bar{\mathbf{c}}/d$;
  $\bar{d} \leftarrow (\bar{d} - \mathbf{c}^\top \bar{\mathbf{c}})/d$;
  $\bar{\mathbf{B}} \leftarrow \bar{\mathbf{B}} - \bar{\mathbf{c}}\mathbf{r}$;
  $\bar{\mathbf{r}} \leftarrow \bar{\mathbf{r}} - \begin{bmatrix}\bar{d} & \bar{\mathbf{c}}^\top\end{bmatrix}\begin{bmatrix}\mathbf{r} \\ \mathbf{B}\end{bmatrix}$;
  $\bar{d} \leftarrow \bar{d}/2$;
**return** $\bar{\mathbf{A}}$;

### D.5 Banded triangular matrix solving

We begin by presenting symbolic differentiation for general equation-solving operations, and then focus on the linear system whose left-hand side is a banded matrix.

**General case** Let $\mathbf{A} \in \mathbb{R}^{M\times M}$ and $\mathbf{B} \in \mathbb{R}^{M\times N}$ be two generic matrices. Let $\mathbf{X} \in \mathbb{R}^{M\times N}$ be the solution to the linear system $\mathbf{AX} = \mathbf{B}$. By taking the differential of $\mathbf{X}$, we obtain the reverse-mode sensitivities for $\mathbf{A}$ and $\mathbf{B}$ (Walter, 2012; Giles, 2008):

$$\mathrm{d}\mathbf{X} = \mathrm{d}(\mathbf{A}^{-1}\mathbf{B}) = \mathrm{d}(\mathbf{A}^{-1})\mathbf{B} + \mathbf{A}^{-1}\,\mathrm{d}\mathbf{B} = -\mathbf{A}^{-1}\,\mathrm{d}\mathbf{A}\mathbf{A}^{-1}\mathbf{B} + \mathbf{A}^{-1}\,\mathrm{d}\mathbf{B}.$$

Equating the terms in the trace of the objective function gradient $\mathrm{tr}(\bar{\mathbf{X}}^\top\,\mathrm{d}\mathbf{X})$, we arrive at the general adjoint relations:

$$\begin{cases}\bar{\mathbf{B}} &= \mathbf{A}^{-\top}\bar{\mathbf{X}},\\ \bar{\mathbf{A}} &= -\mathbf{A}^{-\top}\bar{\mathbf{X}}\mathbf{X}^\top = -\bar{\mathbf{B}}\mathbf{X}^\top,\end{cases}$$

where $\bar{\mathbf{B}}$ can be obtained by solving the adjoint triangular system $\mathbf{A}^\top\bar{\mathbf{B}} = \bar{\mathbf{X}}$.

**Banded case** In IDGPVAE-BSF, the matrix $\mathbf{A}$ is often a banded lower triangular matrix with bandwidth $p$ (e.g., the Cholesky factor $\mathbf{L_K}$). This structure allows for significant computational savings. Therefore, we only need to perform a banded *forward substitution* to compute $\mathbf{X} = \mathbf{A}^{-1}\mathbf{B}$ in $\mathcal{O}(MNp)$ time. When the right-hand side has $M$ columns, as in the computation $\mathbf{L}_{\mathbf{\Sigma}}^{-1}\mathbf{L_K}$, this operation costs $\mathcal{O}(M^2p)$ and can produce a dense lower-triangular output.

Our implementation of operators for IDGPVAE-BSF can handle banded linear systems with structured right-hand sides (r.h.s.). As illustrated in the bottom right of Figure 5, the r.h.s. may exhibit column-wise sparsity (e.g., $\mathbf{K}_{\mathbf{Z}\bar{\mathbf{Z}}_\mathcal{B}}$) or a banded lower triangular structure (e.g., $\mathbf{L_K}$). The jagged profiles observed in the solutions result from sparsity propagation during the triangular solve. Specifically, since $\mathbf{L_\Sigma}$ is lower triangular, forward substitution for the $j$-th column starts at row $i$, where the first non-zero entry of the corresponding r.h.s. column is encountered. Consequently, the "sawtooth" upper boundary of the solution effectively outlines the sparsity pattern of the r.h.s. matrix.

In the backward pass, the sensitivity $\bar{\mathbf{B}}$ can be computed by solving the adjoint system $\mathbf{A}^\top\bar{\mathbf{B}} = \bar{\mathbf{X}}$. Since $\mathbf{A}^\top$ is upper triangular, this step can be efficiently handled via banded *backward substitution*. Furthermore, we only calculate non-zero values of each column of $\bar{\mathbf{B}}$ if $\mathbf{B}$ is column-wise sparse (e.g., $\mathbf{K}_{\bar{\mathbf{Z}}\mathbf{Z}_\mathcal{B}}$) or banded lower triangular (e.g., $\mathbf{L_K}$).

In all cases, we constrain $\bar{\mathbf{A}}$ to share the banded lower-triangular sparsity pattern of $\mathbf{A}$ by applying a fixed mask $\mathcal{S}_L$:

$$\bar{\mathbf{A}} = (-\bar{\mathbf{B}}\mathbf{X}^\top)\odot\mathcal{S}_L,$$

where $\mathcal{S}_L$ is a mask matrix preserving only the $p$ sub-diagonals and the main diagonal. The computational complexity for the backward pass is $\mathcal{O}(MNp)$.

### D.6 Empirical efficiency

In our custom implementation, the CPU operators for the forward passes utilise highly optimised LAPACKE routines (Anderson et al., 1999) where possible. For GPU execution, we instead develop specialised CUDA kernels to support our banded storage format.

To evaluate the practical efficiency of our custom banded operators, we take the Cholesky decomposition operator as an example and benchmark our C++/CUDA implementations against pure PyTorch baselines. For the baselines, we **(1)** use loops to decompose a compactly stored banded matrix iteratively, or **(2)** use native `torch.linalg.cholesky`, which requires expanding the compact banded matrix into a dense representation.

As shown in Table 8, our implementation achieves significant speedup and memory reduction, demonstrating that the efficiency gains are driven by both the exploitation of the banded structure and the integration of low-level kernels.

**Table 8:** Performance comparison for banded Cholesky decomposition ($M$=500, $p$ = 7, batch size = 120, including one backward update).

| IMPLEMENTATION | DEVICE | MEDIAN TIME (IQR) (ms) | PEAK MEMORY (MB) |
|---|---|---|---|
| Pure `PyTorch` (Loop) | Intel CPU | 1431.45 (130.10) | 10096.89 |
| Pure `Pytorch` (Native `cholesky`) | Intel CPU | 1520.71 (23.20) | 461.86 |
| Custom `C++` (`LAPACKE`) | Intel CPU | **7.69** (**0.39**) | **9.80** |
| Pure `PyTorch` (Loop) | NVIDIA L40S | 315.06 (4.74) | 52.90 |
| Pure `Pytorch` (Native `cholesky`) | NVIDIA L40S | 176.81 (8.86) | 1144.41 |
| Custom `CUDA` | NVIDIA L40S | **1.34** (**0.00**) | **6.41** |

**Table 9:** Summary of the relevant models. $N$, $M$, and $H$ denote the numbers of data points, inducing points/features, and nearest neighbours, respectively. $p$ is the bandwidth of the inter-domain matrices. $N_b$ is the mini-batch size. $P$ is the feature dimension of the low-rank representations. $d$ is the dimension of the state variables. $B_r$ is the number of basis functions in the Hilbert-space approximation.

| MODEL | Mini-batching [1] | GP-style posterior $q(\mathbf{Z})$ | Use inducing variables | N.B.[2] prediction | GP complexity |
|---|---|---|---|---|---|
| VAE (Kingma and Welling, 2014) | ✓ | ✗ | / | ✗ | / |
| CVAE (Sohn et al., 2015) | ✓ | ✗ | / | ✗ | / |
| VRNN (Chung et al., 2015) | ✗ | ✗ | / | Implicit[3] | / |
| BRITS (Cao et al., 2018) | ✗ | / | / | Implicit | / |
| LatentODE (Rubanova et al., 2019) | ✗ | ✗ | / | Implicit | / |
| GPVAE-Casale (Casale et al., 2018) | ✗ | ✗ | ✗ | ✗ | $\mathcal{O}(NP^2 + P^3)$ |
| GPVAE-Pearce (Pearce, 2020) | ✗ | ✓ | ✗ | ✗ | $\mathcal{O}(N^3)$ |
| GPVAE-Tridiag (Fortuin et al., 2020) | ✗ | ✗ | ✗ | ✗ | $\mathcal{O}(N^3)$ |
| SVGPVAE (Jazbec et al., 2021) | ✓ | ✓ | ✓ | ✗ | $\mathcal{O}(N_b M^2 + M^3)$ |
| LVAE (Ramchandran et al., 2021) | ✓ | ✗ | ✓ | ✗ | $\mathcal{O}(N_b M^3)$ |
| MGPVAE (Zhu et al., 2023) | ✗ | ✗ | ✗ | ✗ | $\mathcal{O}(d^3 N)$ |
| SGPBAE (Tran et al., 2023) | ✓ | ✗ | ✓ | ✗ | $\mathcal{O}(N_b M^2 + M^3)$ |
| SA-DGBFGP (Balık et al., 2025) | ✓ | ✗ | ✗ | ✗ | $\mathcal{O}(N_b \sum_r B_r)$ |
| NNGPVAE-HPA (Shi et al., 2025b) | ✓ | ✗ | ✗ | ✓ | $\mathcal{O}(N_b H^3)$ |
| NNGPVAE-SPA (Shi et al., 2025b) | ✓ | ✗ | ✗ | ✓ | $\mathcal{O}(N_b H^3)$ |
| IDGPVAE (ours, general) | ✓ | ✓ | ✓ | ✓ | $\mathcal{O}(N_b M^2 + M^3)$ |
| IDGPVAE-VFF (ours, $D$=1) | ✓ | ✓ | ✓ | ✓ | $\mathcal{O}(N_b M^2 + M^3)$ |
| IDGPVAE-BSF (ours, $D$=1) | ✓ | ✓ | ✓ | ✓ | $\mathcal{O}(N_b M p + M^2 p)$ |

[1] Here means mini-batching along the "time" dimension.
[2] Neighbour-based prediction.
[3] Because of the RNN structure in the model.

# E Experimental details and additional results

**Common experimental configuration** Our models are implemented in `PyTorch` (Paszke et al., 2019) and `GPyTorch` (Gardner et al., 2018). Unless otherwise stated, all timings and memory usages are measured on an NVIDIA L40S GPU. The corrupted-frame imputation experiment runs all models on an NVIDIA A100 GPU due to software compatibility constraints from GPVAE-Tridiag. The wall-clock training time per epoch is computed over 20 epochs; we discard the first 5 epochs and report the median of the remaining 15. We summarise the related models in Table 9. The code is available at `https://github.com/shixinxing/IDGPVAE-official`.

We employ the Matérn-$\frac{3}{2}$ kernel for each experiment and use $p$=2 for IDGPVAE-BSF to minimise computational complexity. We choose $[a, b]$ to cover the range of the auxiliary coordinates and, by default, set $[v_0, v_{M+p}] \subseteq [a, b]$ for IDGPVAE-BSF. The NLL is estimated using 20 Monte Carlo samples from the latent variables. Specifically, given $S$ samples from the latent space, i.e., $\mathbf{Z}_s \sim q(\mathbf{Z})$,

the NLL metric is obtained by using the logsumexp operator for numerical stability:

$$\begin{aligned}\text{NLL} &= -\log \int p(\mathbf{Y} \mid \mathbf{Z}) q(\mathbf{Z}) \, \mathrm{d}\mathbf{Z} \approx -\log \frac{1}{S} \sum_s p(\mathbf{Y} \mid \mathbf{Z}_s) \\ &= \log S - \operatorname{logsumexp}_s \log p(\mathbf{Y} \mid \mathbf{Z}_s).\end{aligned}$$

**A note on notational conventions and inducing scale** In the literature on SVGP and inter-domain GP, conventions for counting inducing quantities may differ. To maintain consistency with the canonical literature (Hensman et al., 2018), we define $M$ as the number of inducing frequencies for IDGPVAE-VFF (which yields $2M+1$ inducing variables), and as the number of inducing variables for SVGPVAE and IDGPVAE-BSF. We refer to $M$ as the "inducing scale" or "inducing size".

We justify this setting by noting that, in our amortised IDGPVAE framework, the variational parameters are conditioned on the input data via a shared encoder rather than being learned as free parameters for each inducing point. As a result, the number of learnable parameters of our models does not grow with $M$. In this sense, $M$ in our work controls the resolution of the latent trajectory rather than the dimensionality of the free variational parameter space. Furthermore, our wall-clock time analysis in Section 5.2 reflects the actual computational burden and serves as the ultimate metric for a fair efficiency comparison.

### E.1 Moving ball

This experiment assesses how different latent priors and inducing representations recover latent trajectories from high-dimensional frames. We therefore include VAE as an i.i.d. latent-prior reference and focus the remaining comparison on GPVAE variants. Generic GP-only baselines are not directly applicable because the latent trajectory is unobserved and must be inferred through a nonlinear image encoder.

**Experimental details** All GP models use Matérn-$\frac{3}{2}$ kernels with the trainable lengthscale initialised to 5. We use the same multilayer perceptron (MLP) structures and training settings as in Jazbec et al. (2021) and Pearce (2020). We employ two independent GPs to model the two-dimensional latent variable of the GPVAE. We train all the models for 25000 epochs. The timestamp range is $x \in [0, 29]$. In IDGPVAE-VFF, we fix $[a, b] = [-0.5, 29.5]$. We set the knot range $[v_2, v_M] = [-0.1, 29.1]$ for IDGPVAE-BSF to ensure that every time point is covered by the supports of 3 consecutive B-spline basis functions. We use full-sequence prediction in this experiment.

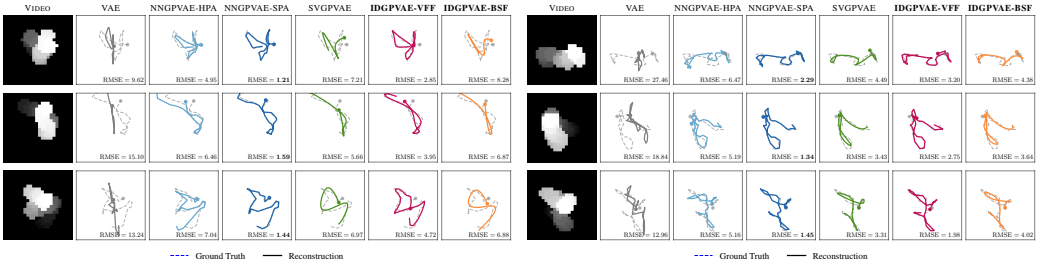
In each training epoch, we generate 35 new videos, each with 30 black-and-white frames from a Matérn-$\frac{3}{2}$ GP, using distinct local seeds. The decoder produces independent Bernoulli distributions for the $32 \times 32$ pixels in each frame. During testing, we compute the mean of the variational posterior over all frames and then apply a least-squares projection to obtain the latent trajectory. We calculate the sum of squared errors on a test batch of 35 videos. The MSE for a single experimental trial is computed by averaging the squared errors across 10 video batches. The RMSE is then the square root of the MSE. The implementation details for the moving ball experiment are summarised in Table 10.

**Table 10:** Implementation configurations for the moving ball experiment.

| Category | Setting | Value |
|---|---|---|
| DATASET | Videos per training epoch | 35 |
| | Frames in each video | 30 |
| | Frame resolution | $32 \times 32$ |
| ARCHITECTURE | Encoder | MLP$(1024, 500, 4)$ |
| | Latent dimensionality ($L$) | 2 |
| | Decoder | MLP$(2, 500, 1024) \rightarrow$ Sigmoid |
| | Activation function | tanh |
| OPTIMISATION | Optimiser | Adam, $\eta = 0.001$ |
| | Training epochs | 25,000 |

**Table 11:** Reconstruction RMSEs of all models in the moving ball experiment, using different inducing scales $M$ or nearest neighbours $H$.

| $M$ or $H$ | 5 | 10 | 15 | 20 |
|---|---|---|---|---|
| VAE (Kingma and Welling, 2014) | | 13.824±4.377 | | |
| GPVAE-Casale (Casale et al., 2018) | | 1.788±0.102 | | |
| GPVAE-Pearce (Pearce, 2020) | | 2.091±0.125 | | |
| SVGPVAE (Jazbec et al., 2021) | 7.814±0.006 | 3.419±0.015 | 2.633±0.063 | 2.866±0.157 |
| NNGPVAE-HPA (Shi et al., 2025b) | 8.506±0.346 | 6.149±0.780 | 4.626±0.307 | 4.016±0.865 |
| NNGPVAE-SPA (Shi et al., 2025b) | 2.142±0.115 | 1.968±0.182 | 1.959±0.091 | 2.047±0.081 |
| IDGPVAE-VFF (ours) | 4.235±0.006 | 2.549±0.013 | **1.499±0.030** | **1.611±0.041** |
| IDGPVAE-BSF (ours) | 8.778±0.007 | 3.636±0.004 | 2.113±0.007 | **1.631±0.015** |




**(a)** Latent trajectory recovery ($M$=5). **(b)** Latent trajectory recovery ($M$=10).

**Figure 7:** Latent trajectory reconstruction on the moving ball dataset using $M$=5/10. The path colours are consistent with the main text. In the low-$M$ regime, NNGPVAE-SPA achieves the lowest RMSE, whereas our models start to capture the main trend. As $M$ increases, the inter-domain features enable IDGPVAE variants to capture finer variations and achieve better reconstruction accuracy.

**Additional results** Table 11 presents reconstruction RMSEs for all models shown in Figure 2a across different inducing scales. Figure 7 visualises the corresponding latent trajectories and additionally includes two NNGPVAE baselines (Shi et al., 2025b). The results indicate that reconstruction improves as the number of inducing features increases, enabling the models to exploit more data information.

### E.2 CMEMS

The CMEMS experiment is designed to directly test the effect of moving from point-evaluated to operator-defined inducing variables and to highlight the accuracy–efficiency trade-off in scalable inference for GP models. We therefore include GP-based baselines in this experiment.

**Experimental details** CMEMS (Lellouche et al., 2021) provides global, gridded ocean reanalysis and forecast products. For our missing-data imputation experiment, we extract three years of hourly oceanic observations (from 25 December 2022 to 25 December 2025) from the Mooring T8S95W record (95°W, 8°S) maintained by the Pacific Marine Environmental Laboratory. The resulting univariate time axis $x \in \mathbb{R}$ indexes time in hours, and the three-dimensional output $\mathbf{y} \in \mathbb{R}^3$ consists of northward and eastward sea-water velocity and sea surface height above the geoid. To simulate missing data, 60% of the data are randomly masked. We then rescale $x$ to the range $[0, 1000]$ and normalise each output dimension of $\mathbf{y}$. The statistics for this normalisation are computed strictly from the observed training data.

The inducing scales $M$ used for each model are shown in the main text (Figure 3) and listed in the appendix (Table 13) with the corresponding total number of model parameters. For all models considered, we initialise the kernel outputscale and lengthscale to 1 and 10, respectively. They are jointly optimised during training. The inducing points in SVGPVAE are shared across all latent dimensions. The neighbour-based prediction for SVGPVAE and our models uses 64 nearest neighbours. The decoder assumes a Gaussian likelihood with fixed noise variance 0.1. To mitigate edge effects in the inter-domain construction, we set $[a, b]$=$[-5, 1005]$ for IDGPVAE-VFF and

**Table 12:** Implementation details for the CMEMS imputation experiment.

| Category | Setting | Value |
|---|---|---|
| DATASET | Sequence length | 26,304 |
| | Missing probability | 60% |
| ARCHITECTURE | Encoder | MLP$(3, 32, 16, 6)$ |
| | Latent dimensionality ($L$) | 3 |
| | Decoder | MLP$(3, 16, 32, 3)$ |
| | Activation function | ReLU |
| OPTIMISATION | Optimiser | Adam, $\eta = 0.001$ |
| | Batch size (time points) | 128 |
| | Training epochs | 1,000 |
| | Trade-off parameter $\beta$ | 0.1 |

$[v_2, v_M]$=$[-1, 1001]$, $p = 2$ for IDGPVAE-BSF. We add a trade-off parameter $\beta$ to the training objective to rebalance the influence of the likelihood and KL terms as in Higgins et al. (2017). The table also includes two NNGPVAE variants with different numbers of nearest neighbours, and we report the better one in the main text. For sampling-based SGPBAE, we set the SGHMC step size to 0.0001. Each inner loop runs 100 SGHMC iterations, followed by 30 stochastic network optimisation iterations. We use 3,000 burn-in iterations and then draw 50 posterior samples with a thinning interval of 400. For MOGP-LMC, we set the number of latent GP functions to 3, matching the number of data channels.

**Additional results** All numerical results used to generate Figure 3 in the main text are reported in Table 13. We also include two NNGPVAE baselines from Shi et al. (2025b). As shown in the table, SGPBAE requires substantially more training time per epoch because it performs inner-loop sampling and optimisation at each iteration. Both SGPBAE and NNGPVAE underperform, possibly because of their mean-field variational posteriors. Figure 8 shows representative imputations from SVGPVAE and IDGPVAE on each data channel. These additional results demonstrate that our inter-domain method consistently outperforms the inducing-point baseline across all channels. Figure 9 further shows how IDGPVAE performance on a 1000-point segment improves as $M$ increases.

**Runtime and memory** To further illustrate how the number of inducing variables translates into both runtime and memory cost in GPVAE models, we provide Figure 3 (right) in the main text, showing training time per epoch and peak GPU memory as functions of the covariance size. For IDGPVAE-VFF, the covariance matrix size is $2M+1$. As shown in the figure, IDGPVAE-VFF achieves shorter training times than SVGPVAE with the same number of inducing variables, owing to its cached cross-covariance terms and block-diagonal covariance structure. For IDGPVAE-VFF, peak memory usage is slightly higher in the low-$M$ regime because we cache the full cross-covariance $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}$ for the entire sequence to reduce repeated kernel evaluations during training. For example, when $M$=128, storing $\mathbf{K}_{\bar{\mathbf{z}}\mathbf{z}}$ in `float64` requires around

$$26{,}304 \times (128 \times 2 + 1) \times 8 \approx 52 \text{ MB},$$

which matches the observed memory gap between SVGPVAE and IDGPVAE-VFF. As $M$ increases, peak memory becomes dominated by dense $M \times M$ linear algebra and autograd buffers, so SVGPVAE shows a steeper growth and eventually exceeds IDGPVAE-VFF. Notably, IDGPVAE-BSF exhibits a much milder increase in both runtime and memory, enabling larger covariance sizes at comparatively low cost.

**Sensitivity to the neighbourhood size $H$** In addition, we investigate how the neighbourhood size $H$ in neighbour-based (NB) prediction affects performance on the CMEMS dataset. To isolate this effect, we fix the inducing size to $M \in \{512, 1024\}$ and sweep $H \in [4, 256]$ for IDGPVAE-BSF. As shown in Figure 10, NB prediction exhibits a clear U-shaped trend in both NLL and RMSE. Very small neighbourhoods (e.g., $H \leqslant 8$) lead to noticeably worse metrics, suggesting insufficient local evidence in this high-frequency dataset. Increasing $H$ initially improves performance, reaching a sweet spot around $H \approx 16 \sim 32$. However, further increasing $H$ gradually degrades performance, and the metrics approach those of the full-sequence (FS) prediction baseline. This decline is consistent with the

intuition behind the "smearing effect": overly large neighbourhoods include distant, less informative observations that reintroduce global conditioning and interfere with local latent reconstruction. Moreover, as $M$ increases, the performance gap between NB and FS predictions narrows, indicating that a higher-resolution latent representation can partially mitigate the smearing effect, thus making the model more robust to $H$.

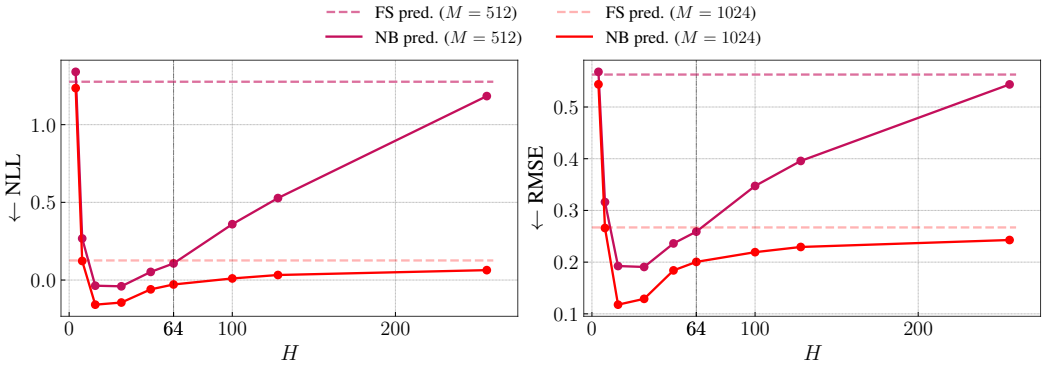



**Figure 10:** NLL and RMSE of neighbour-based prediction on CMEMS as a function of the neighbourhood size $H$. Both metrics exhibit a U-shaped curve: very small neighbourhoods ($H \leqslant 8$) underperform due to insufficient local evidence, while very large neighbourhoods approach the full-sequence baseline as distant, less informative observations induce a smearing effect.

### E.3 Rotated MNIST

Rotated MNIST is our main high-dimensional benchmark for testing whether our approach's gains extend beyond GPVAE-family comparisons. We follow the setup of Fortuin et al. (2020) and Zhu et al. (2023), where individual frames are not assigned discrete instance identifiers, and no ground-truth information from masked frames is used during training. We include conditional and sequential baselines such as CVAE, VRNN, BRITS, and LatentODE, as well as GPVAE variants. GP-only baselines are excluded due to the high dimensionality of the data points. We cannot use VAE for missing-frame generation because it does not leverage auxiliary information. We use the amortised version SA-DGBFGP for the conditional generation task. The RMSE values reported here are computed by rounding the decoder outputs from encoder means at missing locations.

#### E.3.1 Corrupted-Frame Imputation

**Experimental details** The dataset considered in this experiment, commonly referred to as Healing MNIST, is designed to mimic real-world scenarios in which medical data are frequently incomplete (Krishnan et al., 2015). It consists of 50,000 training sequences and 10,000 test sequences. In addition to the standard setting where pixels are Missing Completely At Random (MCAR), we also evaluate two structured missingness mechanisms introduced by Fortuin et al. (2020): positively and negatively temporally correlated missingness (Temporal$^+$ and Temporal$^-$, respectively). Under these mechanisms, if a feature is missing at one time step, it becomes more or less likely, respectively, to be missing at the subsequent time step.

Our experimental setup closely follows Fortuin et al. (2020), in which missing pixels are set to zero during inference and ELBOs are computed only over the observed pixels. All kernel lengthscales are initialised to 2, and timestamps are restricted to the range $[0, 9]$. For IDGPVAE-VFF, we set $[a, b]{=}[-0.5, 9.5]$, while for IDGPVAE-BSF we choose the B-spline knot range $[v_2, v_M]{=}[-0.1, 9.1]$. Given the short sequence length, we employ a full-sequence prediction strategy in this experiment. The complete experimental configuration is summarised in Table 14. We also evaluated CVAE, partially following the design in Casale et al. (2018). Specifically, we preprocess the scaled timestamp $x$ into $[\sin(x), \cos(x)]$ and feed it through an MLP to obtain a time feature, then stack it to the vector after the encoder and before the decoder. In addition, we append the frame mask as an extra channel to the input frames before feeding them into the encoder. We include RNN-based models such as VRNN and BRITS, since the frames are observed at regular time steps. For LatentODE, we clip the gradient norm to 1000 to make it work.

**Table 14:** Implementation details for the pixel imputation task on Rotated MNIST.

| Category | Setting | Value |
|---|---|---|
| DATASET | Training/test sequences | 50,000/10,000 |
| | Sequence length | 10 |
| | Frame resolution | $28 \times 28$ |
| | Missing mechanism | MCAR, Temporal$^+$, Temporal$^-$ |
| ARCHITECTURE | Encoder | Conv$(1,\ 256,\ 3^2) \rightarrow$ Conv$(256,\ 1,\ 3^2) \rightarrow$ MLP$(256, 256, 512)$ |
| | Latent dimensionality ($L$) | 256 |
| | Decoder | MLP$(256, 256, 256, 256, 784) \rightarrow$ Sigmoid |
| | Activation function | ReLU |
| OPTIMISATION | Optimiser | Adam, $\eta = 5 \cdot 10^{-4}$ |
| | Batch size (sequences) | 50 |
| | Training epochs | 100 |
| | Trade-off parameter $\beta$ | 0.2 |

**Additional results** Due to space constraints, the main text reports only a subset of the results; here in the appendix, we provide the complete set of results in Table 15. The imputation results on Temporal$^+$ and Temporal$^-$ Rotated MNIST are also summarised in Table 15. Because the runtimes closely match those in the main text, we omit them in the table. The overall trends are similar to those observed on the MCAR version, with IDGPVAE-VFF consistently achieving the best performance. Additionally, we report the NLL and RMSE results of IDGPVAE-VFF and IDGPVAE-BSF for different values of $M$ in Table 16. As $M$ increases, the IDGPVAE models generally perform better. Figure 11 and Figure 12 present additional qualitative visualisations of the imputation results for other digit sequences.

**Table 17:** Implementation details for the missing-frame generation task on the Rotated MNIST.

| Category | Setting | Value |
|---|---|---|
| DATASET | Training/test sequences | 4,000/1,000 |
| | Sequence length | 100 |
| | Frame resolution | $28 \times 28$ |
| | Missing probability | 60% |
| ARCHITECTURE | Encoder | Conv(1,32,$3^2$)$\rightarrow$Conv(32,64,$3^2$)$\rightarrow$MLP(3136,32) |
| | Latent dimensionality | 16 |
| | Decoder | Deconv(64,32,$3^2$)$\rightarrow$Deconv(32,1,$3^2$)$\rightarrow$Sigmoid |
| | Activation function | ReLU |
| OPTIMISATION | Optimiser | Adam, $\eta = 10^{-3}$ |
| | Batch size (sequences) | 50 |
| | Batch size (frames) | 20 |
| | Training epochs | 300 |
| | Trade-off parameter $\beta$ | 0.2 |

### E.3.2 Missing-Frame Generation

**Experimental details** Following Zhu et al. (2023), this experiment uses Rotated MNIST sequences, where each digit completes two full rotations. We use 4,000 training sequences and 1,000 test sequences, each comprising 100 frames. To simulate missing data, we remove each frame *independently* with probability 0.6, resulting in observation sequences of varying effective lengths.

The encoder–decoder architectures used here also follow Zhu et al. (2023): the encoder is built from convolutional layers, and the decoder from transposed-convolution (deconvolution) layers. All GP kernels are initialised with a lengthscale of 40 and an outputscale of 1. For SVGPVAE, we place the initial inducing points uniformly over the full temporal range. For IDGPVAE-VFF, we set $[a, b]$=$[-5, 104]$, and for IDGPVAE-BSF, we set $[v_2, v_M]$=$[-1, 100]$. All three models use

neighbour-based prediction with $H{=}20$, corresponding to the 20 nearest observed timestamps. More experimental setups are in Table 17. The CVAE architecture also follows Casale et al. (2018), using time features as in the corrupted-frame experiment and leveraging the mean latent representations of observations to generate missing timestamps. For VRNN, we feed the previous hidden state into the prior network, sample $\mathbf{Z}_t$ from the prior network, and then process it with the decoder, as in Zhu et al. (2023). For BRITS, we increase the hidden dimension to 256 and train for 1000 epochs to obtain reasonable performance. For SA-DGBFGP, we follow the preprocessing principles of Riutort-Mayol et al. (2022). Specifically, we standardise $x$ to be centred at zero, choose $\Omega{=}[-1.7, 1.7]$ to contain all values of $x$, and scale the initial kernel lengthscale accordingly to 1.4. We use 32 basis functions for SA-DGBFGP.

**Additional results** The main quantitative results for the missing-frame generation task are summarised in Table 18, where we have added NNGPVAE-HPA and SVGPVAE with 8/24 inducing points. Qualitative examples of the generated frames, illustrating the models' behaviour on long gaps, are presented in Figure 14. We also report the baselines' performance with RBF kernels in Table 19. In this setting, SVGPVAE and NNGPVAE improve but remain below IDGPVAE-VFF. We analyse the effect of the inducing scale $M$ on generation quality in Table 20, which compares SVGPVAE with our two IDGPVAE variants; across most values of $M$, both IDGPVAE models outperform SVGPVAE.

Table 21 compares full-sequence (FS) prediction with neighbour-based (NB) prediction $(H{=}20)$ for IDGPVAE-VFF and IDGPVAE-BSF. In Figure 13, we take IDGPVAE-BSF as an example to further show how performance varies with the neighbourhood size $H$. We observe that the effect of NB prediction depends on the inducing capacity $M$. When $M$ is small, the variational posterior has limited representational fidelity (i.e., a coarse approximation of the latent function), and conditioning on the whole sequence tends to aggregate evidence globally, thereby amplifying temporal oversmoothing. In this regime, using a small $H$ acts as a local conditioning scheme that reduces excessive global averaging. When $M$ is larger, the latent representation is already well resolved, and for smoothly varying dynamics such as Rotated MNIST, FS and NB predictions become similar once $H$ is moderately large. However, too small $H$ can underutilise informative long-range context, leading to inferior reconstruction accuracy. Overall, these trends suggest that, while the optimal regime of $H$ shifts with the inducing capacity $M$, $H$ trades off local sharpness against global context.

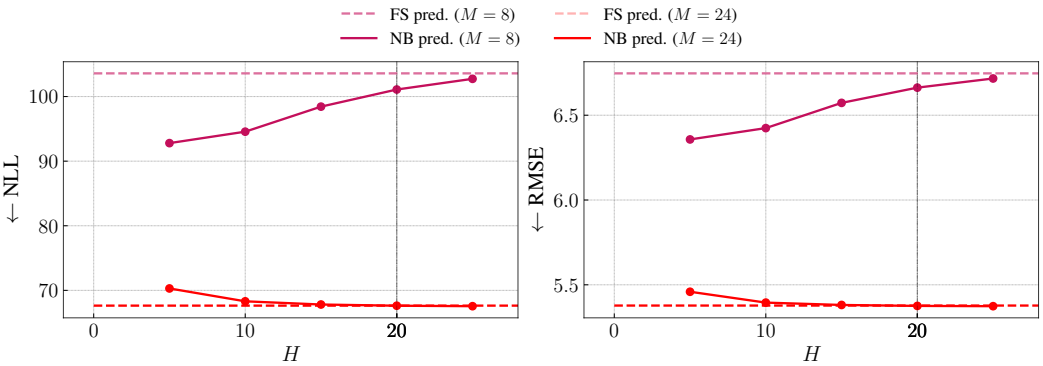



**Figure 13:** Neighbour-based prediction on missing-frame Rotated MNIST as a function of the neighbourhood size $H$.

## F SVGP and inter-domain GP with heterogeneous noise

This section briefly introduces the Sparse Variational Gaussian Process (SVGP; Titsias, 2009) and inter-domain GP (Lázaro-Gredilla and Figueiras-Vidal, 2009). For simplicity, we focus on regression with a Gaussian likelihood. We derive the optimal variational distribution for SVGP with *heterogeneous* Gaussian noise in Appendix F.3. This optimal expression has inspired the variational parameterisation used in Jazbec et al. (2021) and our work. Readers can refer to Hensman et al. (2015) for extensions to non-Gaussian likelihoods.

**GP regression** Without loss of generality, we consider a zero-mean GP $f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$ with inputs $\mathbf{x} \in \mathbb{R}^D$ and a kernel function $k(\cdot, \cdot) : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$. Provided a finite set of inputs $\mathbf{X} = [\mathbf{x}_n]_{n=1}^N \in \mathbb{R}^{N \times D}$, the joint distribution of the corresponding function values $\mathbf{f} = [f(\mathbf{x}_n)]_{n=1}^N \in \mathbb{R}^N$ is given by

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K_{ff}}) = \frac{1}{\sqrt{(2\pi)^N |\mathbf{K_{ff}}|}} \exp\left(-\frac{1}{2}\mathbf{f}^\top \mathbf{K_{ff}}^{-1}\mathbf{f}\right),$$

where $\mathbf{K_{ff}}$ is the $N \times N$ covariance matrix evaluated at all pairs of inputs, i.e., $[\mathbf{K_{ff}}]_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$. Standard GP regression assumes a factorised *Gaussian* likelihood with noise variance $\sigma^2$ over the corrupted outputs $\mathbf{y} = [y_n]_{n=1}^N \in \mathbb{R}^N$, i.e.,

$$p(\mathbf{y}|\mathbf{f}) = \prod_{n=1}^N p(y_n|f_n) = \prod_{n=1}^N \mathcal{N}(y_n|f_n, \sigma^2).$$

The hyperparameters of the GPs (i.e., parameters of the kernel and the likelihood) are learned by maximising the marginal log-likelihood

$$\log p(\mathbf{y}) = \log \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}) \, \mathrm{d}\mathbf{f} = \log \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K_{ff}} + \sigma^2 \mathbf{I})$$

After training, we can perform predictions at test inputs $\mathbf{X}_*$ by first computing the posterior over the corresponding test function values $\mathbf{f}_*$ and then writing the predictive density:

$$p(\mathbf{f}_*|\mathbf{y}) = \int p(\mathbf{f}_*|\mathbf{f}) p(\mathbf{f}|\mathbf{y}) \, \mathrm{d}\mathbf{f}, \quad p(\mathbf{y}_*|\mathbf{y}) = \int p(\mathbf{y}_*|\mathbf{f}_*) p(\mathbf{f}_*|\mathbf{y}) \, \mathrm{d}\mathbf{f}_*,$$

where $p(\mathbf{f}_*|\mathbf{f})$ is the conditional Gaussian distribution derived from the joint Gaussian prior $p(\mathbf{f}_*, \mathbf{f})$. Both training and prediction require $\mathcal{O}(N^3)$ complexity for standard GP regression.

### F.1 SVGP

To alleviate the cubic complexity of exact GP inference, SVGPs leverage a small set of $M (\ll N)$ *inducing points* from the same GP prior to approximate the true posterior (Titsias, 2009; Hensman et al., 2013, 2015). The inducing locations (also known as inducing inputs) and the corresponding inducing variables (also called inducing outputs) are denoted as follows

$$\text{inducing locations} : \mathbf{Z} = [\mathbf{z}_m]_{m=1}^M \in \mathbb{R}^{M \times D}; \quad \text{inducing variables} : \mathbf{u} = [f(\mathbf{z}_m)]_{m=1}^M \in \mathbb{R}^M.$$ [4]

Keeping the GP prior untouched, the joint prior distribution of $\mathbf{f}$ and $\mathbf{u}$ is formulated by

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{u} \end{bmatrix} \sim \mathcal{N}\left( \begin{bmatrix} \mathbf{f} \\ \mathbf{u} \end{bmatrix} | \mathbf{0}, \begin{bmatrix} \mathbf{K_{ff}} & \mathbf{K_{fu}} \\ \mathbf{K_{uf}} & \mathbf{K_{uu}} \end{bmatrix} \right).$$

Therefore, the model can be described by the following distributions:

$$p(\mathbf{y}, \mathbf{f}, \mathbf{u}) = p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\mathbf{u}) p(\mathbf{u}),$$
$$\text{where} \quad p(\mathbf{f}|\mathbf{u}) = \mathcal{N}(\mathbf{f}|\mathbf{K_{fu}}\mathbf{K_{uu}}^{-1}\mathbf{u}, \mathbf{K_{ff}} - \mathbf{K_{fu}}\mathbf{K_{uu}}^{-1}\mathbf{K_{uf}}), \quad p(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{0}, \mathbf{K_{uu}}).$$

The SVGP approach defines the (variational) approximate posterior distribution as

$$q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f}|\mathbf{u}) q(\mathbf{u}), \quad q(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{m}, \mathbf{S}),$$

where $\mathbf{m}$ and $\mathbf{S}$ are variational parameters. All the parameters in SVGP are learned through the minimisation of $\mathrm{KL}[q(\mathbf{f}, \mathbf{u}) \,||\, p(\mathbf{f}, \mathbf{u}|\mathbf{y})]$, which is equivalent to maximising an ELBO to the log-likelihood

$$\begin{aligned} \log p(\mathbf{y}) &\geqslant \int q(\mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{u})}{q(\mathbf{f}, \mathbf{u})} \, \mathrm{d}\mathbf{f} \, \mathrm{d}\mathbf{u} = \int q(\mathbf{u}) p(\mathbf{f}|\mathbf{u}) \log \frac{p(\mathbf{y}|\mathbf{f}) p(\mathbf{u})}{q(\mathbf{u})} \, \mathrm{d}\mathbf{f} \, \mathrm{d}\mathbf{u} \\ &= \int q(\mathbf{u}) \left\{ \int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f}) \, \mathrm{d}\mathbf{f} + \log \frac{p(\mathbf{u})}{q(\mathbf{u})} \right\} \mathrm{d}\mathbf{u} \\ &= \int q(\mathbf{u}) \log \frac{\exp\left(\int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f}) \, \mathrm{d}\mathbf{f}\right) p(\mathbf{u})}{q(\mathbf{u})} \, \mathrm{d}\mathbf{u}, \quad \forall q(\mathbf{u}). \end{aligned} \tag{26}$$

[4] Here, we follow the standard SVGP convention, using $\mathbf{Z}$ for inducing locations and $\mathbf{u}$ for inducing variables. In the main text, however, we reserve $\mathbf{z}/\mathbf{Z}$ for VAE latent variables. To avoid notational conflict, we instead adopt the convention of Snelson and Ghahramani (2005), using an overbar ($\bar{\cdot}$) for inducing quantities in the main text.

The bound reaches equality when $q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f}, \mathbf{u}|\mathbf{y})$. By maximising this bound w.r.t. $q(\mathbf{u})$ or according to the definition of KL divergence, we can obtain the optimal $q^*(\mathbf{u})$ (Titsias, 2009, 2025):

$$q^*(\mathbf{u}) \propto \exp\Big(\underbrace{\int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f}) \, \mathrm{d}\mathbf{f}}_{\text{analytic exponential factor}}\Big) p(\mathbf{u}) \Leftrightarrow \log q^*(\mathbf{u}) = \mathbb{E}_{p(\mathbf{f}|\mathbf{u})} \log p(\mathbf{y}|\mathbf{f}) + \log p(\mathbf{u}) + \text{Const.}$$

The integral in the exponent can be calculated in closed form:

$$\begin{aligned}
\int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f}) \, \mathrm{d}\mathbf{f} &= \int p(\mathbf{f}|\mathbf{u}) \left\{ -\frac{N}{2} \log(2\pi\sigma^2) - \frac{(\mathbf{y}-\mathbf{f})^\top (\mathbf{y}-\mathbf{f})}{2\sigma^2} \right\} \mathrm{d}\mathbf{f} \\
&= -\frac{N}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \int p(\mathbf{f}|\mathbf{u}) \operatorname{tr}\left\{ \mathbf{y}\mathbf{y}^\top - 2\mathbf{y}\mathbf{f}^\top + \mathbf{f}\mathbf{f}^\top \right\} \mathrm{d}\mathbf{f} \\
&= -\frac{N}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \operatorname{tr}\left\{ \mathbf{y}\mathbf{y}^\top - 2\mathbf{y}\mathbf{m}_{\mathbf{f}|\mathbf{u}}^\top + \mathbf{m}_{\mathbf{f}|\mathbf{u}}\mathbf{m}_{\mathbf{f}|\mathbf{u}}^\top + \mathbf{K}_{\mathbf{f}|\mathbf{u}} \right\} \\
&= \log \mathcal{N}\left(\mathbf{y}|\mathbf{m}_{\mathbf{f}|\mathbf{u}}, \sigma^2\mathbf{I}\right) - \frac{1}{2\sigma^2} \operatorname{tr}\left\{ \mathbf{K}_{\mathbf{f}|\mathbf{u}} \right\}
\end{aligned}$$

where $\mathbf{m}_{\mathbf{f}|\mathbf{u}} = \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}$ and $\mathbf{K}_{\mathbf{f}|\mathbf{u}} = \mathbf{K}_{\mathbf{ff}} - \mathbf{Q}_{\mathbf{ff}}$ are the mean and covariance of $p(\mathbf{f}|\mathbf{u})$, respectively. $\mathbf{Q}_{\mathbf{ff}} \coloneqq \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{uf}}$ is the $M$-rank Nyström matrix. Therefore, the optimal $q^*(\mathbf{u})$ can be obtained by

$$\begin{aligned}
q^*(\mathbf{u}) &\propto \exp\left( \int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f}) \, \mathrm{d}\mathbf{f} \right) p(\mathbf{u}) \\
&= \frac{\mathcal{N}\left(\mathbf{y}|\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \sigma^2\mathbf{I}\right) p(\mathbf{u})}{\int \mathcal{N}\left(\mathbf{y}|\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \sigma^2\mathbf{I}\right) p(\mathbf{u}) \, \mathrm{d}\mathbf{u}} = \frac{\mathcal{N}\left(\mathbf{y}|\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \sigma^2\mathbf{I}\right) p(\mathbf{u})}{\mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{Q}_{\mathbf{ff}} + \sigma^2\mathbf{I})} \\
&= \mathcal{N}(\mathbf{u}|\underbrace{(\mathbf{K}_{\mathbf{uu}}^{-1} + \mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{uf}}\sigma^{-2}\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1})^{-1}}_{\boldsymbol{\Sigma}_{\mathbf{u}|\mathbf{y}} = \mathbf{K}_{\mathbf{uu}}\hat{\boldsymbol{\Sigma}}^{-1}\mathbf{K}_{\mathbf{uu}}} \mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{uf}}\sigma^{-2}\mathbf{y}, \boldsymbol{\Sigma}_{\mathbf{u}|\mathbf{y}}) \\
&= \mathcal{N}(\mathbf{u}|\sigma^{-2}\mathbf{K}_{\mathbf{uu}}\hat{\boldsymbol{\Sigma}}^{-1}\mathbf{K}_{\mathbf{uf}}\mathbf{y}, \mathbf{K}_{\mathbf{uu}}\hat{\boldsymbol{\Sigma}}^{-1}\mathbf{K}_{\mathbf{uu}}),
\end{aligned}$$

where $\hat{\boldsymbol{\Sigma}} \coloneqq \mathbf{K}_{\mathbf{uu}} + \sigma^{-2}\mathbf{K}_{\mathbf{uf}}\mathbf{K}_{\mathbf{fu}}$. We do not explicitly write the trace term here since it does not involve $\mathbf{u}$ and has already been included in the denominator.

Once the GP hyperparameters are determined by optimising the ELBO in (26), the posterior predictive distribution is given using optimal $q^*(\mathbf{u})$:

$$\begin{aligned}
q(\mathbf{f}_*|\mathbf{y}) &= \int p(\mathbf{f}_*|\mathbf{f}, \mathbf{u}) p(\mathbf{f}|\mathbf{u}) q^*(\mathbf{u}) \, \mathrm{d}\mathbf{f} \, \mathrm{d}\mathbf{u} \\
&= \int p(\mathbf{f}_*|\mathbf{u}) q^*(\mathbf{u}) \, \mathrm{d}\mathbf{u} \qquad (27) \\
&= \mathcal{N}(\mathbf{f}_* \mid \sigma^{-2}\mathbf{K}_{*\mathbf{u}}\hat{\boldsymbol{\Sigma}}^{-1}\mathbf{K}_{\mathbf{uf}}\mathbf{y}, \mathbf{K}_{**} - \mathbf{K}_{*\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{u}*} + \mathbf{K}_{*\mathbf{u}}\hat{\boldsymbol{\Sigma}}^{-1}\mathbf{K}_{\mathbf{u}*}),
\end{aligned}$$

where $\mathbf{f}_*$ is the collection of random variables at test locations $\mathbf{X}_*$. Equation (27) comes from the GP consistency, with $f(\cdot)|\mathbf{u}$ considered as a conditional GP (Titsias, 2025; Bui and Titsias, 2025). Hence, $p(\mathbf{f}_*|\mathbf{u})$ follows the similar conditional Gaussian form to $p(\mathbf{f}|\mathbf{u})$, just replacing the kernel inputs with $\mathbf{X}_*$.

### F.2 Inter-domain GP

The inter-domain GP (Lázaro-Gredilla and Figueiras-Vidal, 2009; van der Wilk et al., 2020) provides a powerful generalisation of SVGP. Instead of restricting the inducing variable $u = f(\mathbf{z})$ to be a point-evaluation (at the inducing location $\mathbf{z}$) within the same input domain as $\mathbf{x}$, inter-domain GP redefines the inducing variable through a linear operator $\mathcal{L}$ of the original GP $f$. One typical instance of $\mathcal{L}$ is a linear functional integrating $f$ with a deterministic function $\phi$

$$u(\mathbf{z}) = (\mathcal{L}f)(\mathbf{z}) = \int_{\mathbb{R}^D} f(\mathbf{x})\phi(\mathbf{x}, \mathbf{z}) \, \mathrm{d}\mathbf{x}, \quad \mathbf{z} \in \mathbb{R}^{D'}. \qquad (28)$$

The above inducing variable $u(\mathbf{z})$ can be interpreted as a projection of the GP $f$ onto the *feature extraction function* $\phi(\cdot, \mathbf{z})$ over the whole input domain. In practice, $\phi$ can be chosen from a set of basis functions.

$\phi$ specifies the transformed domain in which the pseudo data set lies. If we choose $\phi(\mathbf{x}, \mathbf{z}) \equiv \delta(\mathbf{x}-\mathbf{z})$ where $\delta$ is a Dirac delta function, the inducing-point set is constrained to the original input domain. Hence, there is no longer a transformed space, and the original SVGP model is retrieved. Throughout, we use *(inducing) feature* to refer to the linear operator $\mathcal{L}$ (or equivalently, the feature extraction function $\phi$ in the integral) acting on the prior GP. By selecting an appropriate linear operator $\mathcal{L}$, the resulting variable $u(\mathbf{z})$ is anticipated to provide a more comprehensive understanding of the process than simply relying on point evaluations in the previous SVGP.

Since the transformation (28) is linear, the inter-domain variable $u(\mathbf{Z})$ shares a joint Gaussian distribution with GP $f(\cdot) \sim \mathcal{GP}(\mu(\cdot), k(\cdot,\cdot'))$:

$$\begin{bmatrix} f(\mathbf{X}) \\ u(\mathbf{Z}) \end{bmatrix} \sim \mathcal{N}\left( \begin{bmatrix} \mu(\mathbf{X}) \\ (\mathcal{L}\mu)(\mathbf{Z}) \end{bmatrix}, \begin{bmatrix} k(\mathbf{X},\mathbf{X}) & (k\mathcal{L}')(\mathbf{X},\mathbf{Z}) \\ (\mathcal{L}k)(\mathbf{Z},\mathbf{X}) & (\mathcal{L}k\mathcal{L}')(\mathbf{Z},\mathbf{Z}) \end{bmatrix} \right),$$

where $\mathcal{L}'$ acts on the second argument of the covariance function $k$. Compared to conventional SVGP, inter-domain GP enhances feature extraction by effectively constructing a more informative feature vector $[(\mathcal{L}k)(\cdot, \mathbf{z}_m)]_{m=1}^M \in \mathbb{R}^M$, rather than the standard kernel vector $[k(\cdot, \mathbf{z}_m)]_{m=1}^M \in \mathbb{R}^M$. By plugging these "inter-domain expressions" into the SVGP framework accordingly, inter-domain GPs can leverage the benefits of sparse variational inference.

### F.3 Heterogeneous noise

As our method adopts the pseudo-observation formulation for latent GP, which involves GP regression with *heterogeneous* noise $\mathbf{D} := \operatorname{diag}(\boldsymbol{\sigma}^2)$, we extend the settings of Titsias (2009) and derive a modified optimal $q^*(\mathbf{u})$[5] for completeness:

$$\begin{aligned}
\int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f}) \, \mathrm{d}\mathbf{f} &= \int p(\mathbf{f}|\mathbf{u}) \left\{ -\frac{1}{2}\log(|2\pi\mathbf{D}|) - \frac{(\mathbf{y}-\mathbf{f})^\top \mathbf{D}^{-1}(\mathbf{y}-\mathbf{f})}{2} \right\} \mathrm{d}\mathbf{f} \\
&= -\frac{\log(|2\pi\mathbf{D}|)}{2} - \frac{1}{2}\int p(\mathbf{f}|\mathbf{u}) \operatorname{tr}\left\{ \mathbf{D}^{-1}\mathbf{y}\mathbf{y}^\top - 2\mathbf{D}^{-1}\mathbf{y}\mathbf{f}^\top + \mathbf{D}^{-1}\mathbf{f}\mathbf{f}^\top \right\} \mathrm{d}\mathbf{f} \\
&= -\frac{\log(|2\pi\mathbf{D}|)}{2} \\
&\quad - \frac{1}{2}\operatorname{tr}\left\{ \mathbf{D}^{-1}\mathbf{y}\mathbf{y}^\top - 2\mathbf{D}^{-1}\mathbf{y}\mathbf{m}_{\mathbf{f}|\mathbf{u}}^\top + \mathbf{D}^{-1}\mathbf{m}_{\mathbf{f}|\mathbf{u}}\mathbf{m}_{\mathbf{f}|\mathbf{u}}^\top + \mathbf{D}^{-1}\mathbf{K}_{\mathbf{f}|\mathbf{u}} \right\} \\
&= \log \mathcal{N}\left(\mathbf{y}|\mathbf{m}_{\mathbf{f}|\mathbf{u}}, \mathbf{D}\right) - \frac{1}{2}\operatorname{tr}\left\{ \mathbf{D}^{-1}\mathbf{K}_{\mathbf{f}|\mathbf{u}} \right\},
\end{aligned}$$

where $\mathbf{m}_{\mathbf{f}|\mathbf{u}} = \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}$, $\mathbf{Q}_{\mathbf{ff}} = \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{uf}}$ and $\mathbf{K}_{\mathbf{f}|\mathbf{u}} = \mathbf{K}_{\mathbf{ff}} - \mathbf{Q}_{\mathbf{ff}}$. $\mathbf{m}_{\mathbf{f}|\mathbf{u}}$ and $\mathbf{K}_{\mathbf{f}|\mathbf{u}}$ are the mean and covariance of $p(\mathbf{f}|\mathbf{u})$, respectively. Similarly, the optimal $q^*(\mathbf{u})$ is given by

$$\begin{aligned}
q^*(\mathbf{u}) &\propto \exp\left( \int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f}) \, \mathrm{d}\mathbf{f} \right) p(\mathbf{u}) \\
&= \frac{\mathcal{N}\left(\mathbf{y}|\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \mathbf{D}\right) p(\mathbf{u})}{\int \mathcal{N}\left(\mathbf{y}|\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \mathbf{D}\right) p(\mathbf{u}) \, \mathrm{d}\mathbf{u}} = \frac{\mathcal{N}\left(\mathbf{y}|\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \mathbf{D}\right) p(\mathbf{u})}{\mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{Q}_{\mathbf{ff}} + \mathbf{D})} \\
&= \mathcal{N}(\mathbf{u}| \underbrace{(\mathbf{K}_{\mathbf{uu}}^{-1} + \mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{uf}}\mathbf{D}^{-1}\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1})^{-1}}_{\boldsymbol{\Sigma}_{\mathbf{u}|\mathbf{y}} = \mathbf{K}_{\mathbf{uu}}\hat{\boldsymbol{\Sigma}}^{-1}\mathbf{K}_{\mathbf{uu}}} \mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{uf}}\mathbf{D}^{-1}\mathbf{y}, \boldsymbol{\Sigma}_{\mathbf{u}|\mathbf{y}}) \\
&= \mathcal{N}(\mathbf{u}|\mathbf{K}_{\mathbf{uu}}\hat{\boldsymbol{\Sigma}}^{-1}\mathbf{K}_{\mathbf{uf}}\mathbf{D}^{-1}\mathbf{y}, \mathbf{K}_{\mathbf{uu}}\hat{\boldsymbol{\Sigma}}^{-1}\mathbf{K}_{\mathbf{uu}}),
\end{aligned}$$

where $\hat{\boldsymbol{\Sigma}} = \mathbf{K}_{\mathbf{uu}} + \mathbf{K}_{\mathbf{uf}}\mathbf{D}^{-1}\mathbf{K}_{\mathbf{fu}}$. The variational parameterisations $\mathbf{m}_{\mathcal{B}}$ and $\mathbf{S}_{\mathcal{B}}$ used in Jazbec et al. (2021) and in our work provide stochastic mini-batch estimates of the optimal expression above, obtained by replacing $\mathbf{D}$ with $\frac{|\mathcal{B}|}{N}\boldsymbol{\sigma}_{\mathcal{B}}^2$. In contrast, our method employs operator-dependent covariances. The estimator follows the construction proposed by Evans and Nair (2020) and Jazbec et al. (2021).

[5] Here, $q(\mathbf{u})$ denotes the variational posterior over the inducing variables in sparse GP; it corresponds to the inducing-variable posterior $q(\bar{\mathbf{z}})$ in our approach, where the encoder models a pseudo-likelihood in a latent GP regression.

## G Limitations and future work

**Feature-dependent computational gains** While IDGPVAE provides a unified operator-defined sparse inference framework for GPVAEs, the extent of additional computational benefits depends on the chosen inducing features. When the operators do not induce additional algebraic structure, the method remains in the standard sparse-variational GP complexity regime. The greatest efficiency gains in our current implementation come from the IDGPVAE-BSF instantiation, which induces banded covariance matrices and enables efficient banded linear algebra.

**Inducing-feature design** The IDGPVAE-VFF and IDGPVAE-BSF instantiations studied here are representative examples rather than an exhaustive set of possible operator choices. Developing other linear functionals, such as derivatives (Eriksson et al., 2018; Padidar et al., 2021), convolutional operators (Álvarez and Lawrence, 2011), or domain-specific inducing features, may provide more suitable inductive biases for particular applications and can be a promising direction for future work.

**Auxiliary input dimensionality** Our current instantiations are most effective for one-dimensional continuous auxiliary inputs. Additive kernels provide a practical way to handle multidimensional auxiliary inputs, but such constructions do not necessarily preserve the same banded structure or asymptotic advantages. Recent inter-domain GP techniques that more effectively mitigate the curse of dimensionality (Dutordoir et al., 2020) could be naturally integrated into our framework. Another key avenue for future work is to construct inducing features that capture structure in richer input domains, such as graph (Borovitskiy et al., 2021) or spatio-temporal inputs (Hamelijnck et al., 2021).

**Model architecture** Our study focuses on the scalable GP inference layer while using standard encoder–decoder architectures across most experiments. This design helps isolate the effect of replacing point-evaluated inducing variables with operator-defined inducing variables. A natural extension is to combine the proposed inference framework with richer generative architectures, such as normalising-flow decoders (Rezende and Mohamed, 2015), diffusion-based decoders (Ho et al., 2020), or calibrated neural generative models (Chen and Li, 2023). Inter-domain features have also been explored in compositional GP models (Rudner et al., 2020; Shi et al., 2025a), rather than for amortised inference in GPVAEs.

**Banded-operator implementation** Our custom banded operators are tailored to the compact storage and differentiable solve patterns arising in IDGPVAE-BSF. They are not intended as a fully optimised general-purpose library for banded linear algebra. Even with this specialised but non-exhaustively optimised implementation, IDGPVAE-BSF already yields large practical efficiency gains. Future work could further improve performance through more optimised GPU kernels, alternative storage formats, or integration with mature sparse/banded numerical backends to support a broader range of linear-algebra operators. The proposed framework is independent of the specific implementation and can benefit from any backend supporting the required differentiable banded operations.

## H Licenses

- CMEMS (Lellouche et al., 2021): Copernicus Marine Service Licence Agreement / Terms and Conditions. We acknowledge the source as E.U. Copernicus Marine Service Information (CMEMS), Marine Data Store (MDS)
- MNIST (LeCun et al., 1998): Creative Commons Attribution-Share Alike 3.0 license.
- Rotated MNIST for corrupted frames (Fortuin et al., 2020): MIT license.
- Rotated MNIST for missing frames (Zhu et al., 2023): MIT license.

**Table 13:** Imputation results and computational cost on the long-range CMEMS sequence.

| MODEL | $M$ | #PARAMETERS | NLL (↓) | RMSE (↓) | TIME (s/epoch ↓) |
|---|---|---|---|---|---|
| SGPBAE (Tran et al., 2023) | 256 | 6125 | 4.018±0.176 | 1.352±0.075 | 271.423±3.456 |
| NNGPVAE-HPA (Shi et al., 2025b) | 64 | 1471 | 3.610±0.971 | 1.003±0.002 | **0.901±0.001** |
| | 256 | | 3.458±0.439 | 1.003±0.001 | 18.506±0.002 |
| NNGPVAE-SPA (Shi et al., 2025b) | 64 | 1471 | 3.537±0.655 | 1.003±0.001 | 1.152±0.004 |
| | 256 | | 3.213±0.442 | 1.003±0.002 | 11.611±0.003 |
| MOGP-LMC (van der Wilk et al., 2020) | 256 | ~0.1M | 0.672±0.006 | 0.590±0.002 | 1.790±0.011 |
| | 512 | ~0.4M | 0.793±0.032 | 0.495±0.007 | 2.953±0.015 |
| | 1024 | ~1.5M | 0.080±0.002 | 0.231±0.001 | 7.601±0.008 |
| SVGPVAE (Jazbec et al., 2021) | 128 | 1599 | 0.425±0.055 | 0.370±0.017 | 1.676±0.004 |
| | 256 | 1727 | 0.123±0.007 | 0.263±0.003 | 2.848±0.004 |
| | 384 | 1855 | 0.068±0.008 | 0.242±0.003 | 4.401±0.003 |
| | 512 | 1983 | 0.015±0.006 | 0.218±0.003 | 6.259±0.004 |
| | 640 | 2111 | -0.011±0.004 | 0.207±0.002 | 8.705±0.003 |
| | 768 | 2239 | -0.037±0.002 | 0.195±0.001 | 11.288±0.006 |
| | 896 | 2367 | -0.055±0.005 | 0.186±0.002 | 14.240±0.006 |
| | 1024 | 2495 | -0.073±0.003 | 0.177±0.002 | 18.818±0.004 |
| | 1280 | 2751 | -0.103±0.003 | 0.159±0.002 | 28.423±0.006 |
| IDGPVAE-VFF (ours) | 128 | 1471 | 0.109±0.005 | 0.256±0.002 | 3.057±0.004 |
| | 256 | | -0.026±0.004 | 0.197±0.003 | 6.159±0.007 |
| | 384 | | -0.068±0.005 | 0.177±0.003 | 10.157±0.006 |
| | 512 | | -0.098±0.003 | 0.161±0.002 | 16.622±0.009 |
| | 640 | | **-0.138±0.001** | **0.135±0.000** | 24.157±0.006 |
| IDGPVAE-BSF (ours) | 128 | 1471 | 0.412±0.035 | 0.365±0.010 | 1.188±0.005 |
| | 256 | | 0.252±0.028 | 0.313±0.010 | 1.288±0.005 |
| | 384 | | 0.190±0.017 | 0.291±0.007 | 1.425±0.005 |
| | 512 | | 0.107±0.008 | 0.259±0.003 | 1.586±0.002 |
| | 640 | | 0.061±0.009 | 0.241±0.004 | 1.781±0.003 |
| | 768 | | 0.031±0.006 | 0.228±0.003 | 2.008±0.002 |
| | 896 | | 0.004±0.003 | 0.216±0.001 | 2.234±0.003 |
| | 1024 | | -0.028±0.005 | 0.201±0.003 | 2.499±0.004 |
| | 1280 | | -0.103±0.001 | 0.181±0.001 | 2.982±0.004 |
| | 1792 | | -0.123±0.003 | 0.147±0.002 | 4.185±0.003 |
| | 2304 | | -0.167±0.002 | 0.112±0.001 | 5.474±0.004 |
| | 2816 | | **-0.187±0.001** | **0.093±0.001** | 6.795±0.004 |

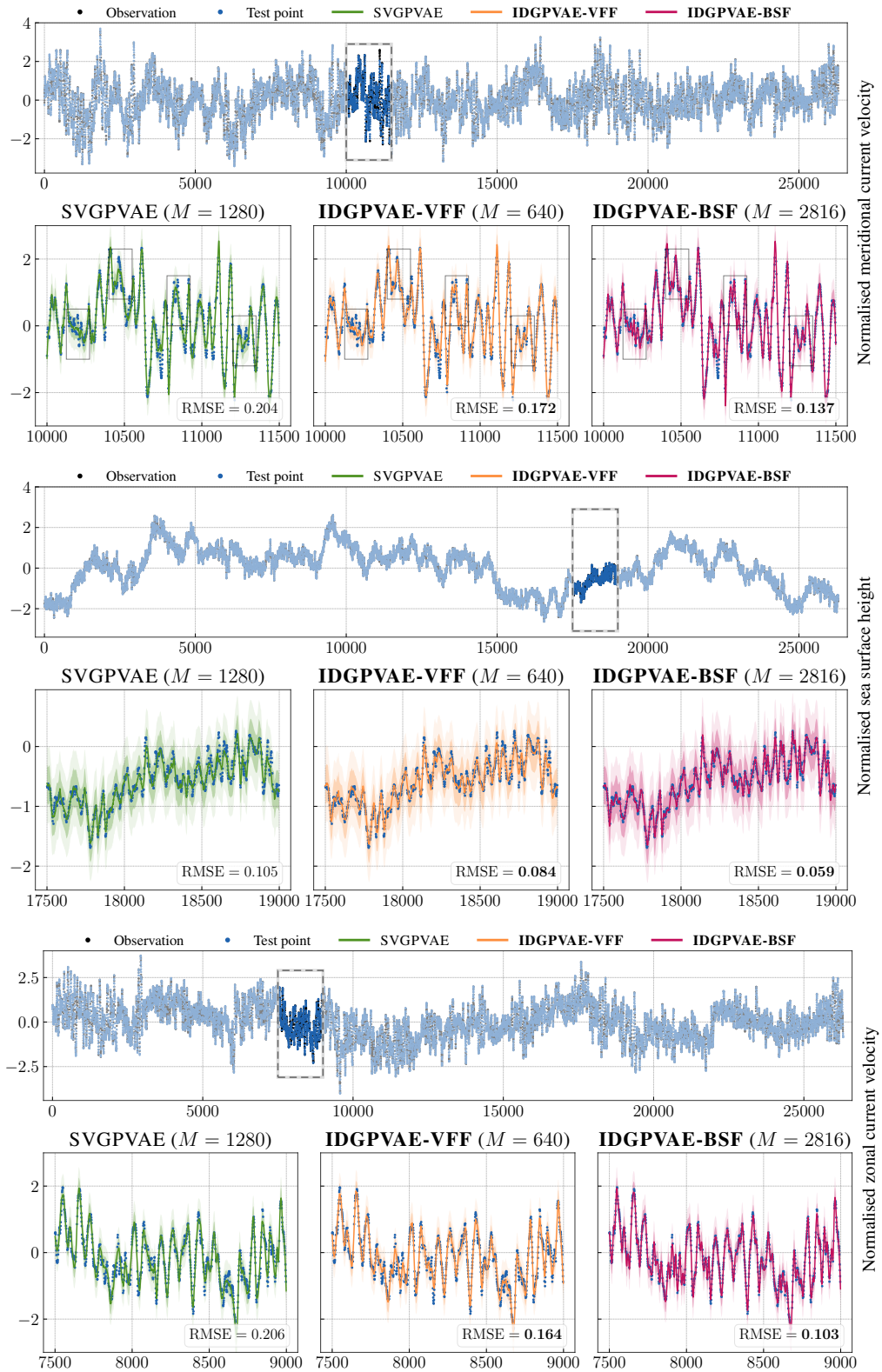



**Figure 8:** Predictions on a 1500-point segment from three channels of the CMEMS sequence. The second row of each panel shows missing points and model imputations, with shading indicating $\pm 1$ and $\pm 2$ standard deviations. All models use the largest inducing scale $M$ reported in this experiment.

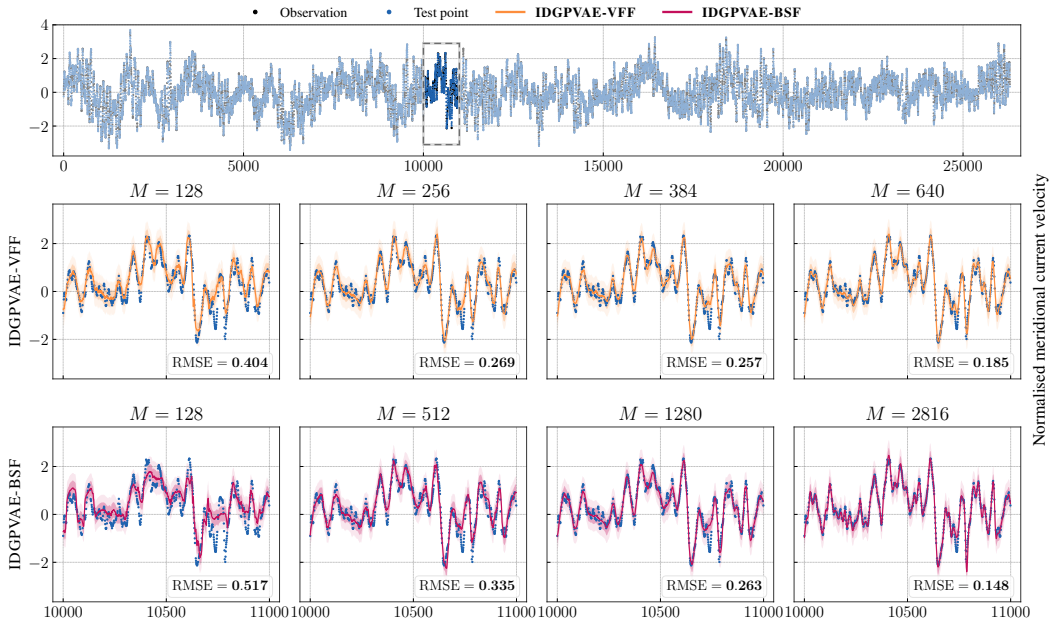



**Figure 9:** Evolution of the performance of IDGPVAE-VFF and IDGPVAE-BSF as the inducing scale $M$ varies on the long-range CMEMS sequence. Increasing $M$ improves the models' ability to capture small variations by providing a higher-resolution latent representation.

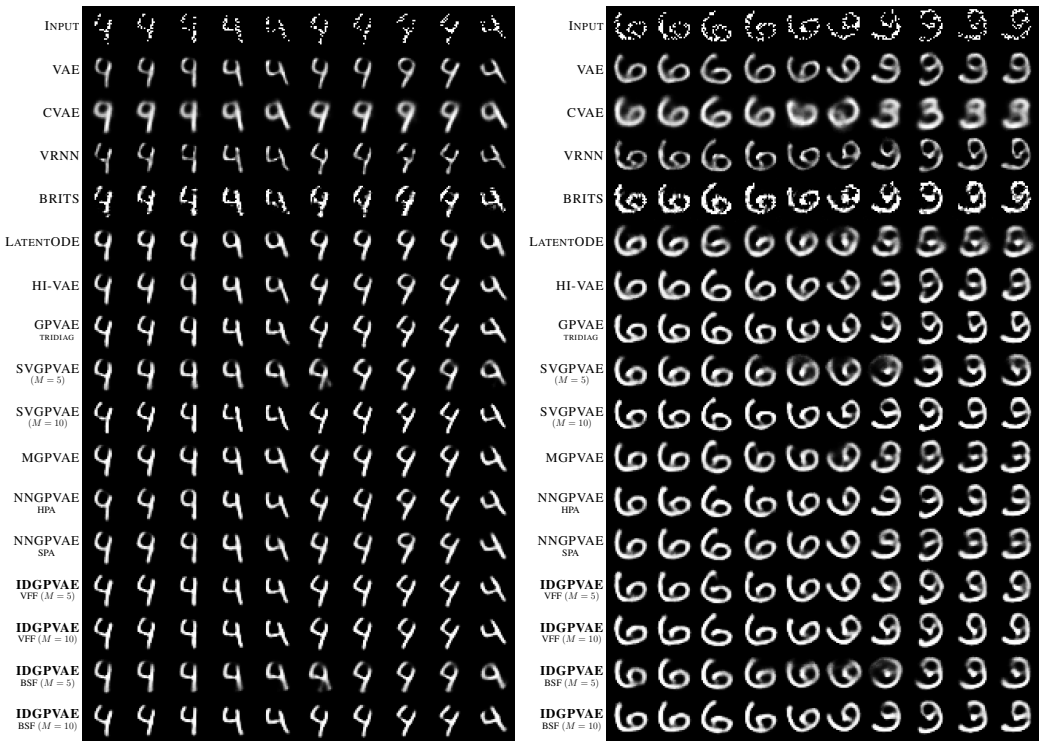



**Figure 11:** Examples of missing-pixel imputation on the MCAR Rotated MNIST dataset. Each panel shows corrupted input frames (top rows) and corresponding reconstructions from all evaluated models. The last four rows in each panel correspond to our proposed IDGPVAE variants. The subset of rows shown in the main text is taken from the left panel.

**Table 15:** Corrupted-frame imputation results on Rotated MNIST under the MCAR (top), Temporal$^+$ (middle), and Temporal$^-$ (bottom) missingness mechanisms.

| Dataset | Model | NLL (↓) | RMSE (↓) | Time (s/epoch ↓)* |
|---|---|---|---|---|
| MCAR | VAE (Kingma and Welling, 2014) | 0.168±0.001 | 0.310±0.002 | **28.599±0.437** |
| | CVAE (Sohn et al., 2015) | 0.143±0.004 | 0.255±0.004 | 39.386±0.619 |
| | VRNN (Chung et al., 2015) | 0.172±0.005 | 0.284±0.004 | 50.972±2.613 |
| | BRITS (Cao et al., 2018) | — | 0.247±0.001 | 47.805±0.342 |
| | LatentODE (Rubanova et al., 2019) | 0.092±0.001 | 0.201±0.001 | 62.651±1.861 |
| | HI-VAE (Nazábal et al., 2020) | 0.097±0.001 | 0.202±0.001 | 30.583±1.104 |
| | GPVAE-Tridiag (Fortuin et al., 2020) | 0.081±0.001 | 0.187±0.001 | 216.797±2.002 |
| | SVGPVAE ($M$=5; Jazbec et al., 2021) | 0.104±0.001 | 0.215±0.001 | 35.062±0.745 |
| | SVGPVAE ($M$=10; Jazbec et al., 2021) | 0.086±0.002 | 0.180±0.001 | 36.199±1.201 |
| | MGPVAE (Zhu et al., 2023) | 0.082±0.001 | 0.187±0.001 | 80.003±2.639 |
| | NNGPVAE-HPA (Shi et al., 2025b) | 0.092±0.000 | 0.195±0.000 | 92.754±0.582 |
| | NNGPVAE-SPA (Shi et al., 2025b) | 0.094±0.001 | 0.199±0.001 | 73.046±1.735 |
| | IDGPVAE-VFF ($M$=5, ours) | **0.072±0.001** | **0.173±0.001** | 33.830±1.428 |
| | IDGPVAE-BSF ($M$=10, ours) | 0.083±0.001 | **0.186±0.001** | 29.314±0.292 |
| Temporal$^+$ | VAE (Kingma and Welling, 2014) | 0.169±0.001 | 0.311±0.002 | — |
| | CVAE (Sohn et al., 2015) | 0.144±0.004 | 0.256±0.004 | — |
| | VRNN (Chung et al., 2015) | 0.161±0.005 | 0.253±0.003 | — |
| | BRITS (Cao et al., 2018) | — | 0.247±0.001 | — |
| | LatentODE (Rubanova et al., 2019) | 0.093±0.001 | 0.202±0.001 | — |
| | HI-VAE (Nazábal et al., 2020) | 0.097±0.001 | 0.202±0.001 | — |
| | GPVAE-Tridiag(Fortuin et al., 2020) | 0.087±0.001 | 0.189±0.001 | — |
| | SVGPVAE ($M$=5, Jazbec et al. (2021)) | 0.107±0.001 | 0.218±0.001 | — |
| | SVGPVAE ($M$=10, Jazbec et al. (2021)) | 0.093±0.001 | 0.185±0.001 | — |
| | MGPVAE (Zhu et al., 2023) | 0.084±0.001 | 0.190±0.001 | — |
| | NNGPVAE-HPA (Shi et al., 2025b) | 0.092±0.001 | 0.195±0.000 | — |
| | NNGPVAE-SPA (Shi et al., 2025b) | 0.095±0.002 | 0.200±0.003 | — |
| | IDGPVAE-VFF ($M$=5, ours) | **0.075±0.001** | **0.178±0.001** | — |
| | IDGPVAE-BSF ($M$=10, ours) | 0.087±0.001 | 0.191±0.001 | — |
| Temporal$^-$ | VAE (Kingma and Welling, 2014) | 0.168±0.001 | 0.309±0.002 | — |
| | CVAE (Sohn et al., 2015) | 0.144±0.000 | 0.256±0.001 | — |
| | VRNN (Chung et al., 2015) | 0.188±0.006 | 0.300±0.003 | — |
| | BRITS (Cao et al., 2018) | — | 0.247±0.002 | — |
| | LatentODE (Rubanova et al., 2019) | 0.091±0.000 | 0.200±0.001 | — |
| | HI-VAE (Nazábal et al., 2020) | 0.096±0.001 | 0.202±0.001 | — |
| | GPVAE-Tridiag (Fortuin et al., 2020) | 0.079±0.001 | 0.181±0.001 | — |
| | SVGPVAE ($M$=5, Jazbec et al. (2021)) | 0.103±0.000 | 0.214±0.000 | — |
| | SVGPVAE ($M$=10, Jazbec et al. (2021)) | 0.085±0.001 | 0.179±0.001 | — |
| | MGPVAE (Zhu et al., 2023) | 0.080±0.001 | 0.186±0.001 | — |
| | NNGPVAE-HPA (Shi et al., 2025b) | 0.092±0.000 | 0.195±0.001 | — |
| | NNGPVAE-SPA (Shi et al., 2025b) | 0.095±0.003 | 0.200±0.003 | — |
| | IDGPVAE-VFF ($M$=5, ours) | **0.071±0.001** | **0.172±0.001** | — |
| | IDGPVAE-BSF ($M$=10, ours) | 0.081±0.000 | 0.184±0.000 | — |

* Training times for Temporal$^+$ and Temporal$^-$ are omitted as they are nearly identical to those under the MCAR mechanism.

**Table 16:** Performance of IDGPVAE-VFF and IDGPVAE-BSF on MCAR Rotated MNIST, reported in terms of NLL and RMSE for different inducing scales $M$.

| $M$ | NLL (↓) | | RMSE (↓) | |
|---|---|---|---|---|
| | IDGPVAE-VFF | IDGPVAE-BSF | IDGPVAE-VFF | IDGPVAE-BSF |
| 3 | 0.087±0.001 | — | 0.195±0.001 | — |
| 5 | 0.072±0.001 | 0.109±0.000 | 0.173±0.001 | 0.220±0.000 |
| 7 | 0.072±0.001 | 0.096±0.000 | 0.174±0.001 | 0.205±0.000 |
| 10 | 0.073±0.001 | 0.083±0.001 | 0.175±0.001 | 0.186±0.001 |

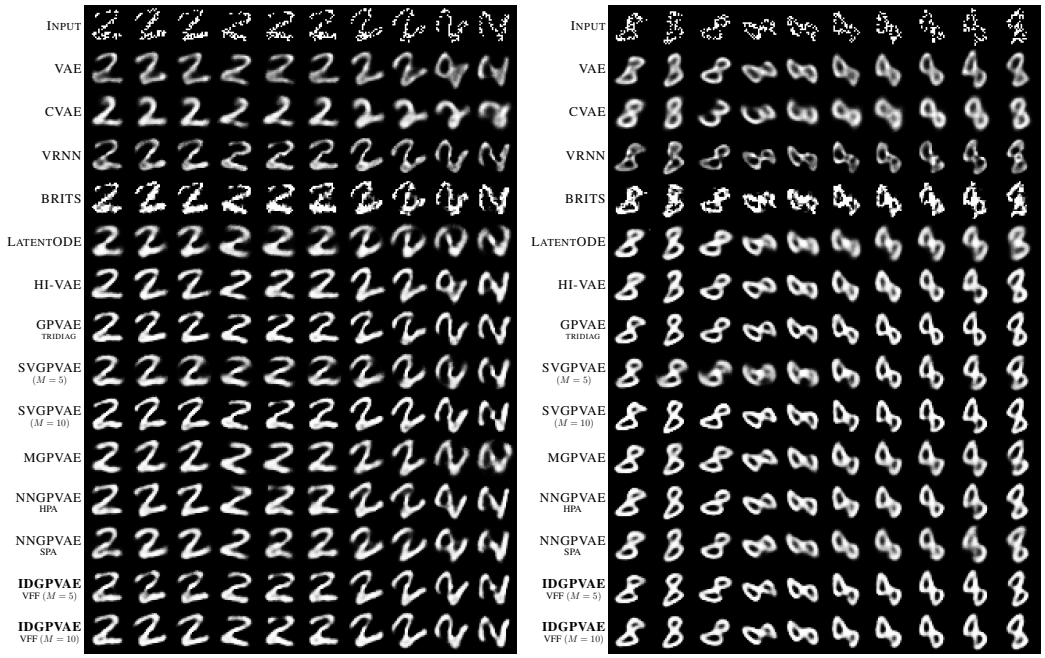



**Figure 12:** Additional examples of missing-pixel imputation on the Temporal$^+$ **(left)** and Temporal$^-$ **(right)** Rotated MNIST dataset, including IDGPVAE-VFF ($M$= 5) and IDGPVAE-BSF ($M$=10). The last two rows show our proposed IDGPVAE variants.

**Table 18:** Full results for Rotated MNIST on the missing-frame generation task.

| MODEL | NLL (↓) | RMSE (↓) | TIME (s/epoch ↓) |
|---|---|---|---|
| CVAE (Sohn et al., 2015) | 288.213±10.422 | 10.925±0.106 | 19.807±0.032 |
| VRNN (Chung et al., 2015) | 82.807±4.362 | 5.984±0.155 | 84.974±0.088 |
| BRITS (Cao et al., 2018) | — | 6.067±0.146 | 54.819±0.053 |
| LatentODE (Rubanova et al., 2019) | 134.131±3.298 | 7.855±0.108 | 46.024±0.061 |
| SVGPVAE ($M$=8; Jazbec et al., 2021) | 89.791±0.482 | 6.255±0.018 | 17.151±0.013 |
| SVGPVAE ($M$=16; Jazbec et al., 2021) | 84.402±0.337 | 5.761±0.008 | 17.175±0.044 |
| SVGPVAE ($M$=24; Jazbec et al., 2021) | 69.800±0.582 | 5.364±0.016 | 17.315±0.025 |
| LVAE (Ramchandran et al., 2021) | 69.729±0.464 | 5.432±0.026 | 67.608±0.278 |
| MGPVAE (Zhu et al., 2023) | 69.116±0.265 | 5.384±0.019 | 39.082±0.081 |
| SA-DGBFGP (Balık et al., 2025) | 88.549±0.992 | 6.130±0.037 | 17.298±0.032 |
| NNGPVAE-HPA (Shi et al., 2025b) | 76.170±0.262 | 5.656±0.005 | 37.944±0.023 |
| NNGPVAE-SPA (Shi et al., 2025b) | 74.770±0.373 | 5.558±0.013 | 36.567±0.047 |
| IDGPVAE-VFF ($M$=8, ours) | **64.826±0.216** | **5.255±0.008** | 17.167±0.030 |
| IDGPVAE-BSF ($M$=24, ours) | **67.792±0.095** | **5.383±0.005** | **17.081±0.020** |

**Table 19:** Additional comparison with RBF-kernel baselines on the Rotated MNIST missing-frames task.

| MODEL | NLL (↓) | RMSE (↓) |
|---|---|---|
| SVGPVAE ($M$=8, Jazbec et al. (2021)) | 92.797±0.987 | 6.350±0.038 |
| SVGPVAE ($M$=24, Jazbec et al. (2021)) | 65.030±0.208 | 5.274±0.009 |
| NNGPVAE-HPA (Shi et al., 2025b) | 68.455±0.203 | 5.420±0.012 |
| NNGPVAE-SPA (Shi et al., 2025b) | 68.402±0.330 | 5.389±0.009 |
| IDGPVAE-VFF($M$=8, ours) | **64.826**±**0.216** | **5.255**±**0.008** |
| IDGPVAE-BSF ($M$=24, ours) | 67.792±0.095 | 5.383±0.005 |

**Table 20:** Additional NLL and RMSE results for varying inducing scales $M$ for missing-frame generation on Rotated MNIST.

| | $M$ | NLL (↓) | | RMSE (↓) | |
|---|---|---|---|---|---|
| | | SVGPVAE | IDGPVAE | SVGPVAE | IDGPVAE |
| IDGPVAE-VFF vs. SVGPVAE | 4 | 113.348±1.397 | **80.050**±**1.558** | 7.096±0.053 | **5.816**±**0.054** |
| | 6 | 99.292±0.563 | **67.371**±**0.671** | 6.601±0.020 | **5.337**±**0.029** |
| | 8 | 89.791±0.482 | **64.826**±**0.216** | 6.255±0.018 | **5.255**±**0.008** |
| | 10 | 87.690±0.562 | **66.855**±**0.231** | 6.103±0.013 | **5.336**±**0.010** |
| IDGPVAE-BSF vs. SVGPVAE | 8 | **89.791**±**0.482** | 101.505±0.548 | **6.255**±**0.018** | 6.676±0.017 |
| | 16 | 84.402±0.337 | **75.715**±**0.142** | 5.761±0.008 | **5.661**±**0.007** |
| | 24 | 69.800±0.582 | **67.792**±**0.095** | **5.364**±**0.016** | 5.383±0.005 |
| | 32 | 71.160±0.475 | **69.564**±**0.178** | **5.425**±**0.008** | 5.461±0.009 |

**Table 21:** Comparison of NLL and RMSE results on the Rotated MNIST dataset, using the full sequence versus the neighbour-based prediction strategy.

| MODEL | $M$ | NLL (↓) | | RMSE (↓) | |
|---|---|---|---|---|---|
| | | Full sequence | Neighbour based | Full sequence | Neighbour based |
| IDGPVAE-VFF | 4 | 100.079±1.504 | **80.050**±**1.558** | 6.552±0.033 | **5.816**±**0.054** |
| | 6 | 82.875±0.655 | **67.371**±**0.671** | 5.827±0.027 | **5.337**±**0.029** |
| | 8 | 66.043±0.294 | **64.826**±**0.216** | 5.306±0.012 | **5.255**±**0.008** |
| | 10 | 66.973±0.255 | **66.855**±**0.231** | 5.344±0.010 | **5.336**±**0.010** |
| IDGPVAE-BSF | 8 | 103.626±0.525 | **101.505**±**0.548** | 6.745±0.015 | **6.676**±**0.017** |
| | 16 | 75.819±0.149 | **75.715**±**0.142** | 5.677±0.008 | **5.661**±**0.007** |
| | 24 | **67.726**±**0.102** | 67.792±0.095 | **5.383**±**0.005** | **5.383**±**0.005** |
| | 32 | **69.311**±**0.192** | 69.564±0.178 | **5.450**±**0.009** | 5.461±0.009 |

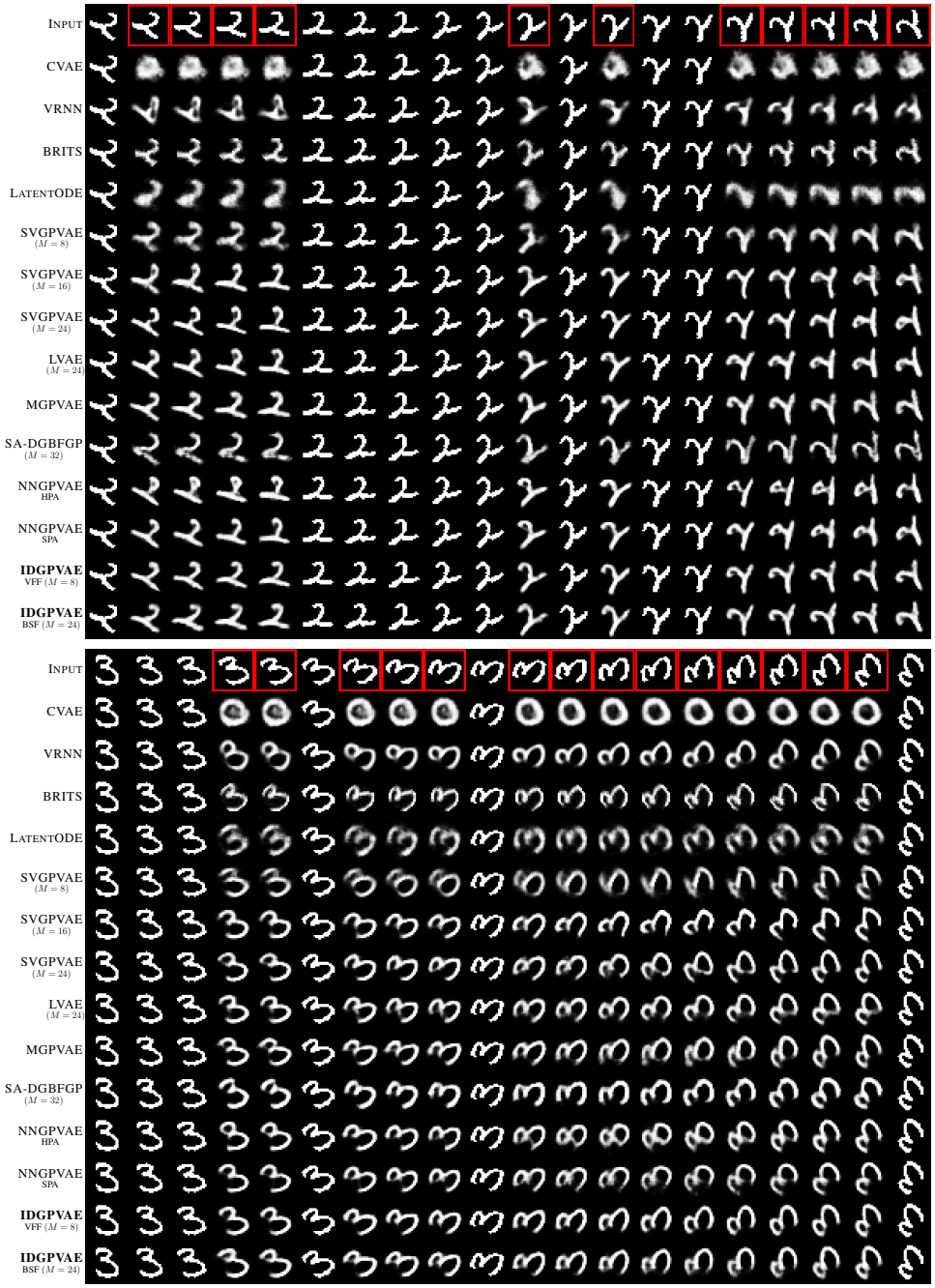



**Figure 14:** Qualitative examples of missing-frame generation on unseen Rotated MNIST sequences. The red boxes represent the missing frames.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the paper's main contributions: a scalable GPVAE inference framework that uses operator-defined inducing variables, the recovery of inducing-point GPVAE as a special case, a cleaner variational objective, and two representative instantiations with Fourier and B-spline features.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses limitations in the Section 6 in the main text and provides a more detailed discussion in the Appendix G.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The main theoretical statement concerns the reduction of IDGPVAE to the standard inducing-point GPVAE objective when the inducing operators are chosen as point evaluations. The required assumptions are stated in Proposition 1, and the full algebraic proof is provided in the Appendix A.3, together with other related identities and derivation details in Appendix A and Appendix F.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper describes the datasets, model variants, baseline choices, evaluation metrics, and timing methodology used in Section 5 and Appendix E. The appendix provides additional experimental details and additional results for each experiment. Results are reported across five random trials, with means and standard deviations.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The paper details experiments, implementation, and data preprocessing for public or reproducible datasets. The link to the implementation code is provided in Appendix E.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The main text describes the experimental tasks, evaluation metrics, baseline groups, and timing protocol needed to interpret the results. The appendix provides full experimental details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All main quantitative results are reported as mean $\pm$ standard deviation over five random trials.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports the hardware used for the experiments at the beginning of Appendix E. We also report wall-clock training time for the experiments, use a consistent timing protocol measured after warm-up, and provide per-epoch training time where computational efficiency is part of the evaluation.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We are not aware of any ethical concerns related to human-subject data, privacy-sensitive information, harmful applications, or violations of anonymity.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: The work is primarily methodological, and any societal impacts are indirect and application-dependent.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: This paper poses no such risks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets used in our experiments are open-source and have been properly referenced in the main text and Appendix H.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release a new dataset, benchmark, or pre-trained model as a new asset at submission time.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments, user studies, or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing, user studies, or research with human subjects, so IRB approval or equivalent review is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method, experiments, and scientific contributions do not involve LLMs as an important, original, or non-standard component.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.