Advances in Scalable Multi-Output Gaussian Process Models

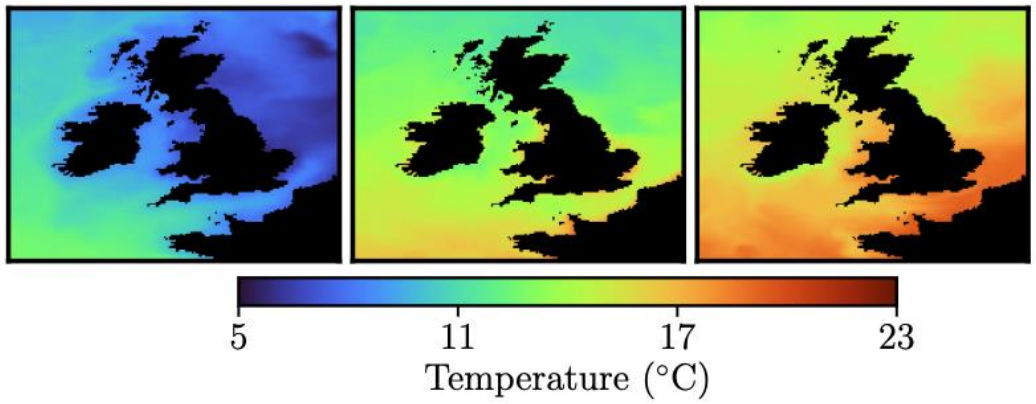
- **Transformed Latent Variable Multi-Output Gaussian Processes (ICML 2026)**

Xiaoyu Jiang

Supervisors: Dr. Mauricio A. Álvarez, Prof. Magnus Rattray

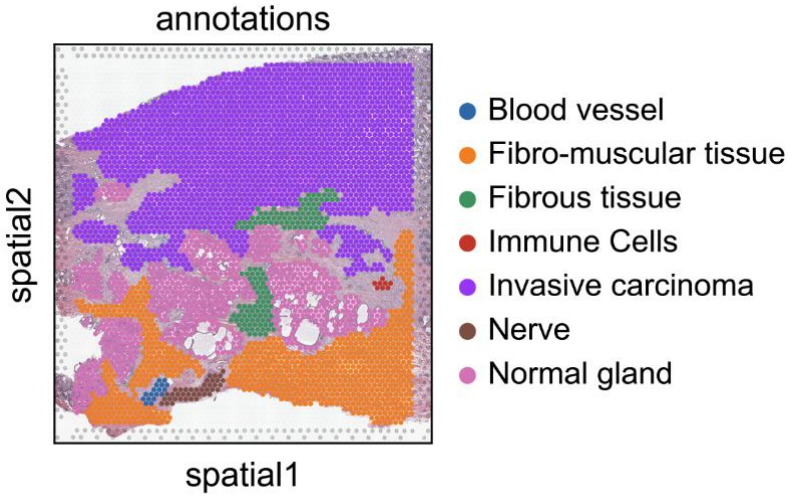
From GPs to MOGPs

- **Gaussian Processes (GPs)** are flexible probabilistic methods to model non-linear functions with **scalar** output, i.e. $y \in \mathbb{R}$.
- In the real world, we are often interested in **vector-valued** functions with P-dimensional observations $\mathbf{y} \in \mathbb{R}^P$. P is potentially large (over 3,000) for many problems.



Climate fields

spatial locations as outputs



Spatial transcriptomics

different genes as different outputs

Setup and Notation

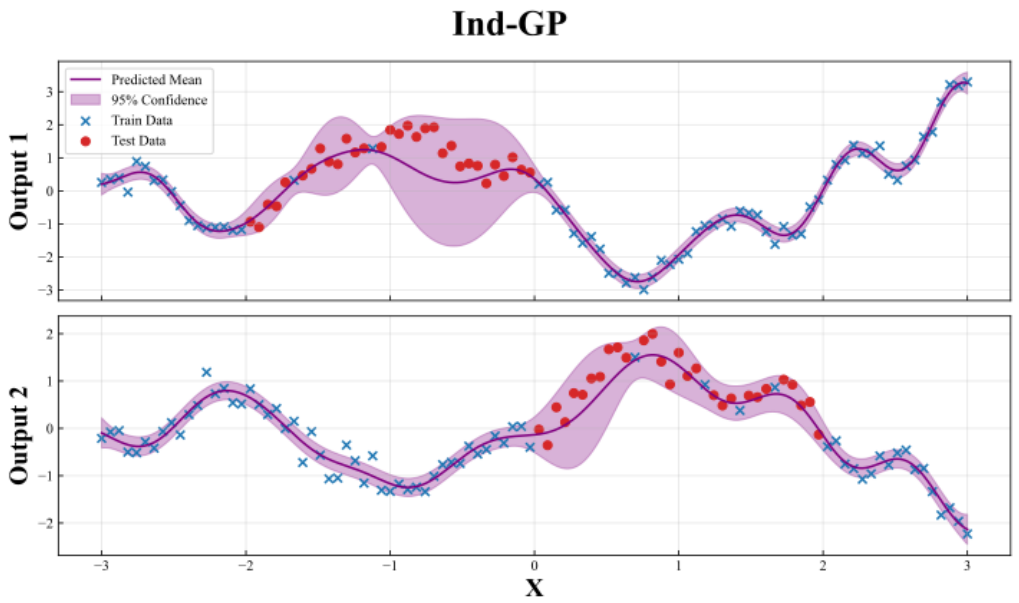
Given a dataset of N input-output pairs $(\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ with $\mathbf{x}_n \in \mathbb{R}^{D_X}$, $\mathbf{y}_n \in \mathbb{R}^P$. We assume the outputs are generated by a **latent function** $f(\mathbf{x})$ with an observation model, such as Gaussian likelihood model.

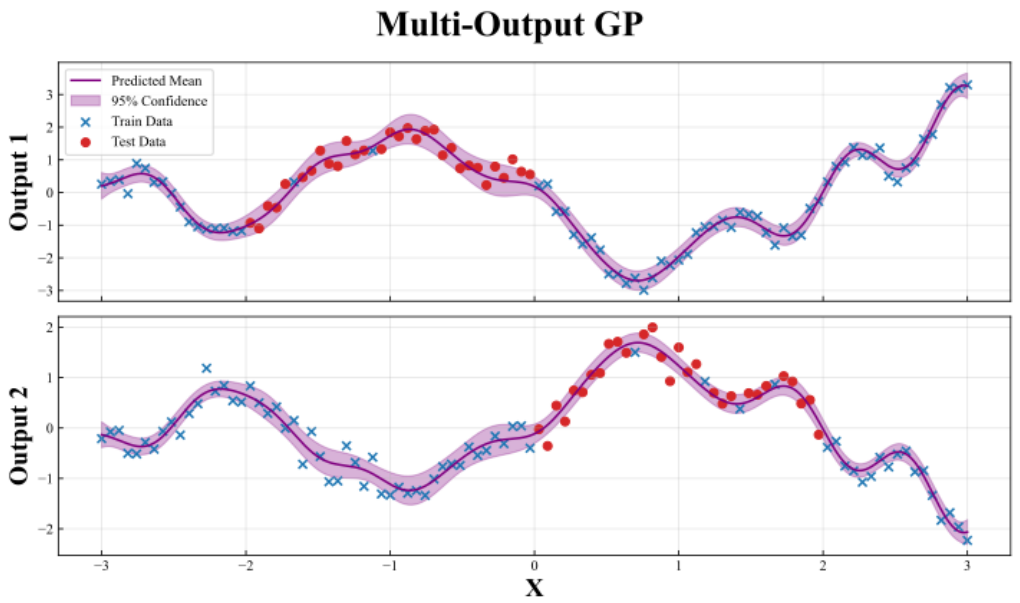
We are interested in modelling the latent function $f(\mathbf{x}) \in \mathbb{R}^P$ using GPs.
One natural approach is to use **independent scalar GPs**, that is

$$f(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_P(\mathbf{x})]^\top$$

where for each p, $f_p(\mathbf{x})$ is a scalar GP, $f_p(\mathbf{x}) \sim \mathcal{GP}(m_p(\cdot), k_p(\cdot, \cdot'))$ and they are all independent.

Independent GPs discard cross-output information





- Independent GPs may fail to provide accurate prediction when observation data is sparse for an output.
- MOGPs can improve prediction by borrowing information across related outputs.

Road Map

- ❑ The Linear Model of Coregionalisation (LMC)
- ❑ Latent Variable MOGP (LVMOGP)
- ❑ Generalised Scalable LVMOGP (GS-LVMOGP)
- ❑ Transformed Latent Variable MOGP (T-LVMOGP)

Road Map

- ❑ The Linear Model of Coregionalisation (LMC)
- ❑ Latent Variable MOGP (LVMOGP)
- ❑ Generalised Scalable LVMOGP (GS-LVMOGP)
- ❑ Transformed Latent Variable MOGP (T-LVMOGP)

The Linear Model of Coregionalisation (LMC)

- LMC expresses outputs as linear combinations of **independent latent GPs**.
- Specifically, the p-th output function $f_p(\mathbf{x})$ is modelled as:

$$f_p(\mathbf{x}_n) = \sum_{q=1}^{Q} \sum_{r=1}^{R_q} \alpha_{p,r}^{(q)} g_r^{(q)}(\mathbf{x}_n),$$

- For each q, $\{g_r^{(q)}(\mathbf{x})\}_{r=1}^{R_q}$ are mutually independent latent GPs with shared input kernel $k_X^{(q)}(\mathbf{x}, \mathbf{x}')$.
- $\{\alpha_{p,r}^{(q)}\}$ are mixing weights.

LMC correlates different output functions by projecting a group of shared independent GPs to the output space.

Journel, A. G. and Huijbregts, C. J. Mining geostatistics. **1976**.

The Linear Model of Coregionalisation (LMC)

- The **cross-covariance** between $f_p(\mathbf{x})$ and $f_{p'}(\mathbf{x}')$ is given by

$$\text{cov}[f_p(\mathbf{x}), f_{p'}(\mathbf{x}')] = \sum_{q=1}^{Q} c_{p,p'}^{(q)} k_X^{(q)}(\mathbf{x}, \mathbf{x}'),$$

- with $c_{p,p'}^{(q)} = \sum_{r=1}^{R_q} \alpha_{p,r}^{(q)} \alpha_{p',r}^{(q)}$

- Let $\mathbf{f} = [\mathbf{f}_1^\top, \ldots, \mathbf{f}_P^\top]^\top \in \mathbb{R}^{PN}$, be the vectorised latent function values with $\mathbf{f}_p = f_p(\mathbf{X})$. The covariance matrix for $\mathbf{f}$ is:

$$\text{cov}[\mathbf{f}, \mathbf{f}] = \sum_{q=1}^{Q} \mathbf{A}^{(q)} \mathbf{A}^{(q)\top} \otimes \mathbf{K}_{\mathbf{XX}}^{(q)} = \sum_{q=1}^{Q} \mathbf{C}^{(q)} \otimes \mathbf{K}_{\mathbf{XX}}^{(q)},$$

where $\mathbf{A}^{(q)} \in \mathbb{R}^{P \times R_q}$ collects $\alpha_{p,r}^{(q)}$, and $\mathbf{C}^{(q)} \in \mathbb{R}^{P \times P}$ contains $c_{p,p'}^{(q)}$.

What LMC assumes

The coregionalisation matrix $\mathbf{C}^{(q)} \in \mathbb{R}^{P \times P}$ models cross-output covariances.

LMC implies that $\mathbf{C}^{(q)}$ has rank at most R_q, it is **low-rank** when $R_q < P$.

It is derived from a **linear kernel** applied to the latent output embeddings. (the rows of $\mathbf{A}^{(q)}$)

The covariance matrix is of a sum-of-separable structure.

The naïve implementation of exact LMC has $\mathcal{O}(N^3P^3)$ computational complexity for log likelihood and posterior computation.

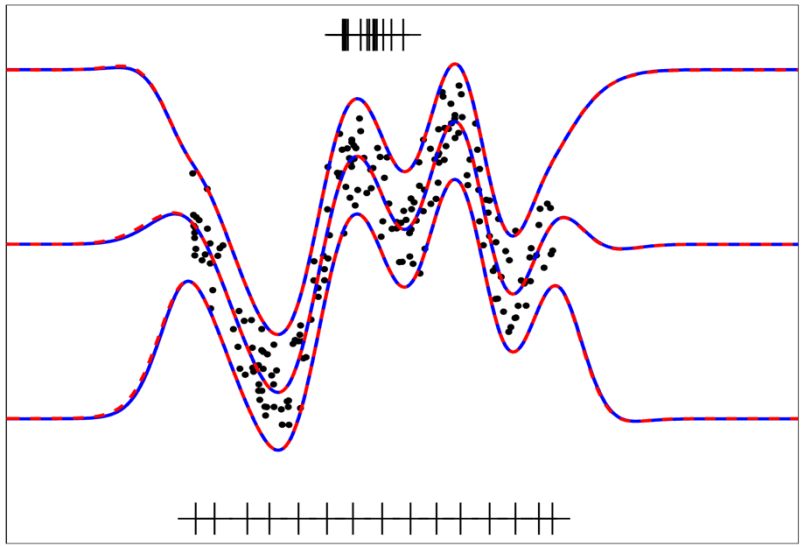
Sparse Variational GP: why inducing points?

Exact GP demands the **cholesky** and **log determinant** computation of an $N \times N$ matrix.
Standard dense training costs $\mathcal{O}(N^3)$ time complexity and $\mathcal{O}(N^2)$ memory complexity.

SVGP introduces M inducing locations, with $M \ll N$:

$$\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^{M},\ \mathbf{z}_i \in \mathbb{R}^{Dx},\ \mathbf{u} = \{f(\mathbf{z}_i)\}_{i=1}^{M}.$$

Think of $\mathbf{u}$ as a small set of uncertain "anchor" function values.



Credit: Titsias, M.

The inducing locations are learned, and they function as **"summary"** of the dataset.

All observations still contribute during the training, but the covariance matrix to deal with is now $M \times M$.

Titsias, M. Variational learning of inducing variables in sparse Gaussian processes. **AISTATS, 2009**.
Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. **UAI, 2013**.

Sparse Variational GP: learning and prediction.

1. Learn a Gaussian Variational distribution over the inducing values

$$p(\mathbf{u}) = \mathcal{N}(\mathbf{u} \mid \mathbf{0}, \mathbf{K_{uu}}),\ q(\mathbf{u}) = \mathcal{N}(\mathbf{u} \mid \mathbf{m}, \mathbf{S}).$$

2. Propagate uncertainty through the GP conditional

$$q(f(\mathbf{x})) = \int p(f(\mathbf{x}) \mid \mathbf{u}) q(\mathbf{u}) d\mathbf{u} = \mathcal{N}(f(\mathbf{x}) \mid m(\mathbf{x}), v(\mathbf{x})).$$

$$m(\mathbf{x}) = k(\mathbf{x}, \mathbf{Z}) k(\mathbf{Z}, \mathbf{Z})^{-1} \mathbf{m}$$

$$v(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - k(\mathbf{x}, \mathbf{Z}) k(\mathbf{Z}, \mathbf{Z})^{-1} k(\mathbf{Z}, \mathbf{x}) + k(\mathbf{x}, \mathbf{Z}) k(\mathbf{Z}, \mathbf{Z})^{-1} \mathbf{S} k(\mathbf{Z}, \mathbf{Z})^{-1} k(\mathbf{Z}, \mathbf{x})$$

3. Train by maximising the evidence lower bound (ELBO)

$$\mathcal{L} = \sum_{n=1}^{N} \mathbb{E}_{q(f(\mathbf{x}_n))}[\log p(y_n \mid f(\mathbf{x}_n))] - \mathrm{KL}(q(\mathbf{u}) \,||\, p(\mathbf{u}))$$

$$\approx \frac{N}{N_b} \sum_{n \in \mathcal{B}} \mathbb{E}_{q(f(\mathbf{x}_n))}[\log p(y_n \mid f(\mathbf{x}_n))] - \mathrm{KL}(q(\mathbf{u}) \,||\, p(\mathbf{u}))$$

Titsias, M. Variational learning of inducing variables in sparse Gaussian processes. **AISTATS, 2009**.
Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. **UAI, 2013**.

Sparse Variational LMC

- ❑ Place SVGPs [Titsias, 2009; Hensman et al., 2013] in the latent space of the LMC.
- ❑ Perform mini-batch training across both inputs and outputs.

- For each latent function $g_r^{(q)}(\mathbf{x})$, we introduce inducing points $\mathbf{Z}_r^{(q)} \in \mathbb{R}^{M \times D_X}$ and inducing variables $\mathbf{u}_r^{(q)} \in \mathbb{R}^M$. The variational distributions for inducing variables are **fully factorised** across latent GPs:

$$q(\mathbf{u}_r^{(q)}) = \mathcal{N}(\mathbf{u}_r^{(q)} \mid \mathbf{m}_r^{(q)}, \mathbf{S}_r^{(q)})$$

- where $\mathbf{m}_r^{(q)} \in \mathbb{R}^M$ and $\mathbf{S}_r^{(q)} \in \mathbb{R}^{M \times M}$.

Titsias, M. Variational learning of inducing variables in sparse Gaussian processes. **AISTATS, 2009**.
Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. **UAI, 2013**.
Van der Wilk, M., Dutordoir, V., John, S., Artemev, A., Adam, V., and Hensman, J. A framework for interdomain and multioutput Gaussian processes. arXiv preprint arXiv:2003.01115, **2020**.

Making Predictions in the SV-LMC (Step1)

Step1: Make predictions in the latent space.
Step2: Project them to the output space.

Given test input $\mathbf{x}_* \in \mathbb{R}^{D_X}$, the predictive distribution $q(g_r^{(q)}(\mathbf{x}_*))$ is:

$$q(g_r^{(q)}(\mathbf{x}_*)) = \int p(g_r^{(q)}(\mathbf{x}_*) \mid \mathbf{u}_r^{(q)}) q(\mathbf{u}_r^{(q)}) d\mathbf{u}_r^{(q)} = \mathcal{N}(m_r^{(q)}(\mathbf{x}_*), v_r^{(q)}(\mathbf{x}_*)),$$

$$m_r^{(q)}(\mathbf{x}_*) = k_q(\mathbf{x}_*, \mathbf{Z}_r^{(q)}) k_q(\mathbf{Z}_r^{(q)}, \mathbf{Z}_r^{(q)})^{-1} \mathbf{m}_r^{(q)}$$

$$v_r^{(q)}(\mathbf{x}_*) = k_q(\mathbf{x}_*, \mathbf{x}_*) - k_q(\mathbf{x}_*, \mathbf{Z}_r^{(q)}) k_q(\mathbf{Z}_r^{(q)}, \mathbf{Z}_r^{(q)})^{-1} k_q(\mathbf{Z}_r^{(q)}, \mathbf{x}_*)$$
$$+ k_q(\mathbf{x}_*, \mathbf{Z}_r^{(q)}) k_q(\mathbf{Z}_r^{(q)}, \mathbf{Z}_r^{(q)})^{-1} \mathbf{S}_r^{(q)} k_q(\mathbf{Z}_r^{(q)}, \mathbf{Z}_r^{(q)})^{-1} k_q(\mathbf{Z}_r^{(q)}, \mathbf{x}_*).$$

Follows standard SVGP prediction. The predictive posterior is fully factorised across latent GPs.

Titsias, M. Variational learning of inducing variables in sparse Gaussian processes. **AISTATS, 2009.**
Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. **UAI, 2013.**

Making Predictions in the SV-LMC (Step2)

Step1: Make predictions in the latent space.
Step2: Project them to the output space.

Recall that $f_p(\mathbf{x}_*) = \sum_{q=1}^{Q} \sum_{r=1}^{R_q} \alpha_{p,r}^{(q)} g_r^{(q)}(\mathbf{x}_*)$, denote $q(f_p(\mathbf{x}_*)) = \mathcal{N}(f_p(\mathbf{x}_*) \mid m_p(\mathbf{x}_*), v_p(\mathbf{x}_*))$

- where $m_p(\mathbf{x}_*) = \sum_{q=1}^{Q} \sum_{r=1}^{R_q} \alpha_{p,r}^{(q)} m_r^{(q)}(\mathbf{x}_*)$
- and $v_p(\mathbf{x}_*) = \sum_{q=1}^{Q} \sum_{r=1}^{R_q} (\alpha_{p,r}^{(q)})^2 v_r^{(q)}(\mathbf{x}_*)$

ELBO of SV-LMC with Stochastic Optimisation

$$\text{ELBO: } \mathcal{L} = \sum_{p=1}^{P}\sum_{n=1}^{N} \mathbb{E}_{q(f_p(\mathbf{x}_n))}\left[\log p(y_{n,p} \mid f_p(\mathbf{x}_n))\right] - \sum_{q=1}^{Q}\sum_{r=1}^{R_q} \mathrm{KL}[q(\mathbf{u}_r^{(q)}) \,||\, p(\mathbf{u}_r^{(q)})]$$

For the expected log likelihood, we employ **stochastic mini-batch training**. At each iteration, we sample input and output mini-batches uniformly, denoted as $\mathcal{B}_N$ and $\mathcal{B}_P$ with cardinality N_b and P_b.

$$\tilde{\mathcal{L}} = \frac{NP}{N_b P_b} \sum_{p\in\mathcal{B}_P}\sum_{n\in\mathcal{B}_N} \mathbb{E}_{q(f_p(\mathbf{x}_n))}\left[\log p(y_{n,p} \mid f_p(\mathbf{x}_n))\right] - \sum_{q=1}^{Q}\sum_{r=1}^{R_q} \mathrm{KL}[q(\mathbf{u}_r^{(q)}) \,||\, p(\mathbf{u}_r^{(q)})]$$

Road Map

- ❑ The Linear Model of Coregionalisation (LMC)
- ❑ Latent Variable MOGP (LVMOGP)
- ❑ Generalised Scalable LVMOGP (GS-LVMOGP)
- ❑ Transformed Latent Variable MOGP (T-LVMOGP)

From LMC to LVMOGP

Recall that in LMC, when $Q = 1$, $\mathbf{C} = \mathbf{A}\mathbf{A}^\top$. The elements in the coregionalisation matrix are the inner product between rows of $\mathbf{A}$.

Key ideas of the LVMOGP,

- ❑ Associate each output p with a latent embedding $\mathbf{h}_p$. Collectively denoted as $\mathbf{H} \in \mathbb{R}^{P \times D_H}$.
- ❑ Generalise linear kernel to any valid kernel function, and $\mathbf{C} = k_H(\mathbf{H}, \mathbf{H})$.
- ❑ Take Bayesian treatment to $\mathbf{H}$, that is introducing the prior and variational posterior distributions to these latent variables.

A general positive-definite kernel $k_H(\cdot, \cdot')$ can model "nonlinear" cross-output covariances.

Dai, Z., Alvarez, M., and Lawrence, N. Efficient modelling of latent information in supervised learning using Gaussian processes. **NeurIPS, 2017**.

Generative model of LVMOGP

$$p(\mathbf{H}) = \prod_{p=1}^{P} p(\mathbf{h}_p) = \prod_{p=1}^{P} \mathcal{N}(\mathbf{h}_p \mid \mathbf{0}, \mathbb{I}_{D_H})$$

$$p(\mathbf{f} \mid \mathbf{X}, \mathbf{H}) = \mathcal{N}(\mathbf{f} \mid \mathbf{0}, \mathbf{K}_{\mathbf{f},\mathbf{f}})$$

$$\mathbf{K}_{\mathbf{f},\mathbf{f}} = k_H(\mathbf{H}, \mathbf{H}) \otimes k_X(\mathbf{X}, \mathbf{X})$$

$$p(\mathbf{Y} \mid \mathbf{f}) = \prod_{n=1}^{N} \prod_{p=1}^{P} p(y_{n,p} \mid f_p(\mathbf{x}_n))$$

Bayesian latent variables capture uncertainty about output embeddings and cross-output relationships.

Dai, Z., Alvarez, M., and Lawrence, N. Efficient modelling of latent information in supervised learning using Gaussian processes. **NeurIPS, 2017**.

Sparse Variational Inference for LVMOGP

Place inducing points in both the **input space** and the **latent space**, which are denoted as

$$\mathbf{Z}_X = \{\mathbf{z}_X^{(1)}, \mathbf{z}_X^{(2)}, ..., \mathbf{z}_X^{(M_X)}\}, \qquad \mathbf{Z}_H = \{\mathbf{z}_H^{(1)}, \mathbf{z}_H^{(2)}, ..., \mathbf{z}_H^{(M_H)}\}$$

The inducing variables $\mathbf{u} \in \mathbb{R}^{M_X M_H}$ follows the same GP prior:

$$p(\mathbf{u}) = \mathcal{N}(\mathbf{u} \mid \mathbf{0}, \mathbf{K}_{\mathbf{u},\mathbf{u}}), \qquad \mathbf{K}_{\mathbf{u},\mathbf{u}} = k_H(\mathbf{Z}_H, \mathbf{Z}_H) \otimes k_X(\mathbf{Z}_X, \mathbf{Z}_X)$$

The conditional distribution of $\mathbf{f}$ given $\mathbf{u}$ and $\mathbf{H}$ is:

$$p(\mathbf{f} \mid \mathbf{u}, \mathbf{H}) = \mathcal{N}(\mathbf{f} \mid \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}, \mathbf{K}_{\mathbf{f},\mathbf{f}} - \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}})$$

where $\mathbf{K}_{\mathbf{f},\mathbf{u}} = k_H(\mathbf{H}, \mathbf{Z}_H) \otimes k_X(\mathbf{X}, \mathbf{Z}_X)$.

The variational distribution for $\mathbf{u}$ is $q(\mathbf{u}) = \mathcal{N}(\mathbf{u} \mid \mathbf{m}_\mathbf{u}, \mathbf{S}_\mathbf{u})$, with $\mathbf{m}_\mathbf{u} \in \mathbb{R}^{M_X M_H}$ and $\mathbf{S}_\mathbf{u} \in \mathbb{R}^{M_X M_H \times M_X M_H}$.
The variational distribution for $\mathbf{H}$ is $q(\mathbf{H}) = \prod_{p=1}^{P} q(\mathbf{h}_p) = \prod_{p=1}^{P} \mathcal{N}(\mathbf{h}_p \mid \mathbf{m}_p, \mathbf{S}_p)$, with $\mathbf{m}_p \in \mathbb{R}^{D_H}$, $\mathbf{S}_p \in \mathbb{R}^{D_H \times D_H}$.
The ELBO is

$$\mathcal{L} = \mathbb{E}_{p(\mathbf{f}|\mathbf{u},\mathbf{H})q(\mathbf{u})q(\mathbf{H})}\left[\log p(\mathbf{Y} \mid \mathbf{f})\right] - \mathrm{KL}[q(\mathbf{u}) \,||\, p(\mathbf{u})] - \mathrm{KL}[q(\mathbf{H}) \,||\, p(\mathbf{H})].$$

Dai, Z., Alvarez, M., and Lawrence, N. Efficient modelling of latent information in supervised learning using Gaussian processes. **NeurIPS, 2017**.

What changes from LMC to LVMOGP?

Model	Output representation	Cross-output covariance	Structure	Large P
LMC	Rows of $\mathbf{A}$	Linear kernel	**Sum of separable**	**Possible, via SV-LMC**
LVMOGP	**Latent variables $\mathbf{H}$**	**Any valid kernel**	Single separable	Limited

GS-LVMOGP extends LVMOGP to accommodate:

(1) Sum of separable covariance structure, and

(2) Stochastic optimisation with mini-batching across both inputs and outputs.

Road Map

- ❑ The Linear Model of Coregionalisation (LMC)
- ❑ Latent Variable MOGP (LVMOGP)
- ❑ Generalised Scalable LVMOGP (GS-LVMOGP)
- ❑ Transformed Latent Variable MOGP (T-LVMOGP)

GS-LVMOGP adds multiple separable components

- Instead of a single latent variable per output in LVMOGP, they propose the use of $Q \geq 1$ latent variables, denoted as $\mathbf{H} = \{\mathbf{H}^{(q)}\}_{q=1}^{Q}$ and $\mathbf{H}^{(q)} \in \mathbb{R}^{P \times D_H}$ contains the latent vector $\mathbf{h}_p^{(q)} \in \mathbb{R}^{D_H}$ for the p-th output in the q-th group.

- We have Q groups of inducing points in the latent space, that is $\mathbf{Z}_H = \{\mathbf{Z}_H^{(q)}\}_{q=1}^{Q}$, with $\mathbf{Z}_H^{(q)} \in \mathbb{R}^{M_H \times D_H}$.

- Introduce Q kernel functions for both the input space and the latent space, which are:

$$\{k_H^{(q)}\}_{q=1}^{Q}, \qquad \{k_X^{(q)}\}_{q=1}^{Q}.$$

The input locations $\mathbf{X}$ and input-space inducing locations $\mathbf{Z}_X$ are shared across components.

Jiang, X., Georgaka, S., Rattray, M., and ´Alvarez, M. A. Scalable multi-output Gaussian processes with stochastic variational inference. **TMLR, 2025**.

Covariance matrices

$$\mathbf{K}_{\mathbf{f},\mathbf{f}} = \sum_{q=1}^{Q} k_H^{(q)}(\mathbf{H}^{(q)}, \mathbf{H}^{(q)}) \otimes k_X^{(q)}(\mathbf{X}, \mathbf{X}),$$

$$\mathbf{K}_{\mathbf{f},\mathbf{u}} = \sum_{q=1}^{Q} k_H^{(q)}(\mathbf{H}^{(q)}, \mathbf{Z}_H^{(q)}) \otimes k_X^{(q)}(\mathbf{X}, \mathbf{Z}_X),$$

$$\mathbf{K}_{\mathbf{u},\mathbf{u}} = \sum_{q=1}^{Q} k_H^{(q)}(\mathbf{Z}_H^{(q)}, \mathbf{Z}_H^{(q)}) \otimes k_X^{(q)}(\mathbf{Z}_X, \mathbf{Z}_X).$$

The input locations $\mathbf{X}$ and input-space inducing locations $\mathbf{Z}_X$ are shared across components.

Jiang, X., Georgaka, S., Rattray, M., and ´Alvarez, M. A. Scalable multi-output Gaussian processes with stochastic variational inference. **TMLR, 2025**.

More expressive covariance structure

$$\mathbf{K}_{\mathbf{f},\mathbf{f}} = \sum_{q=1}^{Q} k_H^{(q)}(\mathbf{H}^{(q)}, \mathbf{H}^{(q)}) \otimes k_X^{(q)}(\mathbf{X}, \mathbf{X}).$$

q = 1	Output geometry $(k_H^{(1)}, \mathbf{H}^{(1)})$	×	Short lengthscale input pattern
q = 2	Output geometry $(k_H^{(2)}, \mathbf{H}^{(2)})$	×	Long lengthscale input pattern
	… …		… …
q = Q	Output geometry $(k_H^{(Q)}, \mathbf{H}^{(Q)})$	×	Medium lengthscale input pattern

sum the components

Different kernels to act on different latent spaces to capture various types of correlations among outputs → more expressive

Sparse Variational Inference for GS-LVMOGP

The variational distribution for $\mathbf{H}$ is

$$q(\mathbf{H}) = \prod_{q=1}^{Q} q(\mathbf{H}^{(q)}) = \prod_{q=1}^{Q} \prod_{p=1}^{P} q(\mathbf{h}_p^{(q)}) = \prod_{q=1}^{Q} \prod_{p=1}^{P} \mathcal{N}(\mathbf{h}_p^{(q)} \mid \mathbf{m}_p^{(q)}, \mathbf{S}_p^{(q)}),$$

where $\mathbf{m}_p^{(q)} \in \mathbb{R}^{D_H}, \mathbf{S}_p^{(q)} \in \mathbb{R}^{D_H \times D_H}$.

The variational distribution for $\mathbf{u}$ is

$$q(\mathbf{u}) = \mathcal{N}(\mathbf{u} \mid \mathbf{m}_\mathbf{u}, \mathbf{S}_\mathbf{u}), \qquad \mathbf{m}_\mathbf{u} \in \mathbb{R}^{M_X M_H}, \qquad \mathbf{S}_\mathbf{u} \in \mathbb{R}^{M_X M_H \times M_X M_H}.$$

The variational distribution for $\mathbf{f}$ given $\mathbf{H}$ is

$$\begin{aligned} q(\mathbf{f} \mid \mathbf{H}) &= \int p(\mathbf{f} \mid \mathbf{u}, \mathbf{H}) q(\mathbf{u}) d\mathbf{u} \\ &= \mathcal{N}(\mathbf{f} \mid \mathbf{K}_{\mathbf{f},\mathbf{u}} \mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{m}_\mathbf{u}, \mathbf{K}_{\mathbf{f},\mathbf{f}} - \mathbf{K}_{\mathbf{f},\mathbf{u}} \mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{K}_{\mathbf{u},\mathbf{f}} + \mathbf{K}_{\mathbf{f},\mathbf{u}} \mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{S}_\mathbf{u} \mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{K}_{\mathbf{u},\mathbf{f}}). \end{aligned}$$

Jiang, X., Georgaka, S., Rattray, M., and ´Alvarez, M. A. Scalable multi-output Gaussian processes with stochastic variational inference. **TMLR, 2025**.

ELBO of GS-LVMOGP with Stochastic Optimisation

$$\mathcal{L} = \underbrace{\mathbb{E}_{q(\mathbf{f}|\mathbf{H})q(\mathbf{H})}\left[\log p(\mathbf{Y} \mid \mathbf{f})\right]}_{\mathcal{T}_1} - \underbrace{\mathrm{KL}[q(\mathbf{u}) \,||\, p(\mathbf{u})]}_{\mathcal{T}_2} - \underbrace{\mathrm{KL}[q(\mathbf{H}) \,||\, p(\mathbf{H})]}_{\mathcal{T}_3}.$$

$$\mathcal{T}_1 = \sum_{p=1}^{P}\sum_{n=1}^{N} \mathbb{E}_{q(f_{n,p}|\mathbf{H}_p)q(\mathbf{H}_p)}[\log p(y_{n,p} \mid f_{n,p})]$$

$$\approx \frac{NP}{N_b P_b} \sum_{p\in\mathcal{B}_P}\sum_{n\in\mathcal{B}_N} \mathbb{E}_{q(f_{n,p}|\mathbf{H}_p)q(\mathbf{H}_p)}[\log p(y_{n,p} \mid f_{n,p})]$$

$$\approx \frac{NP}{N_b P_b} \sum_{p\in\mathcal{B}_P}\sum_{n\in\mathcal{B}_N} \frac{1}{J}\sum_{j=1}^{J} \mathbb{E}_{q(f_{n,p}|\mathbf{H}_p^{(j)})}[\log p(y_{n,p} \mid f_{n,p})]$$

where $\mathbf{H}_p^{(j)}, j \in \{1, 2, ..., J\}$ are samples from $q(\mathbf{H}_p)$.

$$\mathcal{T}_3 = \sum_{p=1}^{P}\sum_{q=1}^{Q} \mathrm{KL}[q(\mathbf{h}_p^{(q)}) \,||\, p(\mathbf{h}_p^{(q)})]$$

$$\approx \frac{P}{P_b} \sum_{p\in\mathcal{B}_P}\sum_{q=1}^{Q} \mathrm{KL}[q(\mathbf{h}_p^{(q)}) \,||\, p(\mathbf{h}_p^{(q)})]$$

Two sources of stochasticity: (1) mini-batching observed input-output pairs, and (2) Monte Carlo sampling of $\mathbf{H}$.

Jiang, X., Georgaka, S., Rattray, M., and ´Alvarez, M. A. Scalable multi-output Gaussian processes with stochastic variational inference. **TMLR, 2025**.

Compare LMC, LVMOGP and GS-LVMOGP

Model	Output representation	Cross-output covariance	Structure	Large P
LMC	rows of $\mathbf{A}$	Linear kernel	Sum of separable	Possible, via SV-LMC
LVMOGP	latent variables $\mathbf{H}$	Any valid kernel	Single separable	Limited
GS-LVMOGP	**Q groups of latent variables $\mathbf{H} = \{\mathbf{H}^{(q)}\}_{q=1}^{Q}$**	**Any valid kernel**	**Sum of separable**	**Yes***

*The computational cost of each iteration does not increase with P.

GS-LVMOGP offers scalable training updates within a structured covariance family.
Can we gain more modelling flexibility while keeping training updates tractable?

Road Map

- ❑ The Linear Model of Coregionalisation (LMC)
- ❑ Latent Variable MOGP (LVMOGP)
- ❑ Generalised Scalable LVMOGP (GS-LVMOGP)
- ❑ Transformed Latent Variable MOGP (T-LVMOGP)

Can we achieve better trade-off between scalability and expressiveness?

- Existing MOGPs typically resort to structural assumptions for scaling up to large output space.
  - **Low rank:** OILMM [Bruinsma et al., 2020]
  - **Grid structure:** SGPRN [Li et al., 2020]
  - **Sum of separable:** SV-LMC [Van der Wilk et al., 2020], GS-LVMOGP [Jiang et al., 2025]

- Scalability techniques have been well established in single-output GP literature. SVGPs are well established, recent works [Titsias, 2025; Bui et al., 2025] have proposed tighter variational bounds.

💡 Propose a multi-output deep kernel without an explicitly prescribed low-rank, grid, or finite sum-of-separable structure.
💡 Reformulates MOGPs as scalar GPs defined on a learned embedding space.
💡 Typical scalable GP techniques can be seamlessly reused to improve the scalability with respect to the number of outputs.

Van der Wilk, M., Dutordoir, V., John, S., Artemev, A., Adam, V., and Hensman, J. A framework for interdomain and multioutput Gaussian processes. arXiv preprint arXiv:2003.01115, **2020**.
Bruinsma, W., Perim, E., Tebbutt, W., Hosking, S., Solin, A., and Turner, R., Scalable exact inference in multi-output Gaussian processes. **ICML, 2020**.
Li, S., Xing, W., Kirby, R. M., and Zhe, S., Scalable Gaussian process regression networks. **IJCAI, 2020**.
Jiang, X., Georgaka, S., Rattray, M., and Álvarez, M., Scalable multi-output Gaussian processes with stochastic variational inference. **TMLR, 2025**.
Titsias, M. New bounds for sparse variational Gaussian processes. **ICML, 2025**.
Bui, T. D., Ashman, M., and Turner, R. E. Tighter sparse variational Gaussian processes. **TMLR, 2025**.
Jiang X, Shi X, Georgaka S, Rattray M, Álvarez MA. Transformed Latent Variable Multi-Output Gaussian Processes. **ICML, 2026**.

Key idea: learn an embedding space

- GS-LVMOGP learns covariances by placing kernel functions in both input and latent spaces.
- We propose to learn **an embedding space** (through a trainable function $\Phi_\theta(\cdot)$), where a base kernel can capture flexible **cross-output** and **cross-input** dependencies.
- Similar to LVMOGP, place a latent variable to each output that act as output representation, denoted as $\mathbf{H} \in \mathbb{R}^{P \times D_H}$.

$$(\mathbf{x}_n, \mathbf{h}_p) \xmapsto{\Phi_\theta} \tilde{\mathbf{x}}_{n,p} \in \mathbb{R}^{D_T},$$

$$\operatorname{cov}\left[f_p(\mathbf{x}_n), f_{p'}(\mathbf{x}_{n'})\right] = k_{\text{base}}\left(\tilde{\mathbf{x}}_{n,p}, \tilde{\mathbf{x}}_{n',p'}\right).$$

The embedding space induces a more expressive cross-output structure.

Jiang X, Shi X, Georgaka S, Rattray M, Álvarez MA. Transformed Latent Variable Multi-Output Gaussian Processes. **ICML, 2026**.

Coupling $\mathbf{x}_n$ with $\mathbf{h}_p$ allows non-separable covariance

same output pair (p, p′)

Context $\mathbf{x}_i$

strong covariance

Context $\mathbf{x}_j$

weak covariance

$$k_{\text{base}}(\Phi_\theta([\mathbf{x}_n, \mathbf{h}_p]), \Phi_\theta([\mathbf{x}_n, \mathbf{h}_{p'}])) \text{ can vary with } \mathbf{x}_n$$

T-LVMOGP can represent cross-output relationships that change across the input domain.

Jiang X, Shi X, Georgaka S, Rattray M, Álvarez MA. Transformed Latent Variable Multi-Output Gaussian Processes. **ICML, 2026**.

LVMOGP (Q=1) kernel is recovered by a blockwise linear embedding

Recall that for LVMOGP with Q=1:

$$k_{\mathrm{LVMOGP}}([\mathbf{x}, \mathbf{h}_p], [\mathbf{x}', \mathbf{h}_{p'}]) = k_X(\mathbf{x}, \mathbf{x}')k_H(\mathbf{h}_p, \mathbf{h}_{p'})$$

$$\Phi_\theta(\mathbf{x}, \mathbf{h}) = [\mathbf{x} \oslash \boldsymbol{\ell}_X; \mathbf{h} \oslash \boldsymbol{\ell}_H], \oslash \text{ denotes element-wise division}$$

$$\boldsymbol{\ell}_X \in \mathbb{R}_{>0}^{D_X} \text{ and } \boldsymbol{\ell}_H \in \mathbb{R}_{>0}^{D_H}$$

ARD-RBF kernel $\Rightarrow$ $k_{\mathrm{base}}(\Phi_\theta(\mathbf{x}, \mathbf{h}_p), \Phi_\theta(\mathbf{x}', \mathbf{h}_{p'})) = k_X(\mathbf{x}, \mathbf{x}')k_H(\mathbf{h}_p, \mathbf{h}_{p'})$

Restrict $\Phi_\theta(\cdot, \cdot)$ to linear, block-separated scaling $\Rightarrow$ **Recover the Q=1 LVMOGP kernel exactly**

T-LVMOGP extends LVMOGP by replacing the block-separated linear embedding with a learned nonlinear map.

GS-LVMOGP (Q > 1) kernel is a block-additive special case

Recall that for GS-LVMOGP with Q > 1:

$$k([\mathbf{x}, \mathbf{h}_p], [\mathbf{x}', \mathbf{h}_{p'}]) = \sum_{q=1}^{Q} k_X^{(q)}(\mathbf{x}, \mathbf{x}') k_H^{(q)}(\mathbf{h}_p^{(q)}, \mathbf{h}_{p'}^{(q)})$$

$$\Phi_\theta(\mathbf{x}, \mathbf{h}_p) = \text{concat}\left(\Phi_\theta^{(1)}(\mathbf{x}, \mathbf{h}_p^{(1)}), \Phi_\theta^{(2)}(\mathbf{x}, \mathbf{h}_p^{(2)}), ..., \Phi_\theta^{(Q)}(\mathbf{x}, \mathbf{h}_p^{(Q)})\right)$$

where $\Phi_\theta^{(q)}(\mathbf{x}, \mathbf{h}) = [\mathbf{x} \oslash \boldsymbol{\ell}_X^{(q)}; \mathbf{h} \oslash \boldsymbol{\ell}_H^{(q)}]$, $\oslash$ denotes element-wise division

$$k_{\text{base}}(\mathbf{z}, \mathbf{z}') = \sum_{q=1}^{Q} k^{(q)}(\mathbf{z}^{(q)}, \mathbf{z}'^{(q)}), \ \mathbf{z}^{(q)} = \Phi_\theta^{(q)}(\mathbf{x}, \mathbf{h}_p^{(q)})$$

Each additive component factorises as in the Q=1 case, giving the sum-of-separable kernel structure.

A general neural map $\Phi_\theta(\cdot)$ and stationary base kernel $k_{\text{base}}(\cdot, \cdot')$ can represent non-separable kernels outside the chosen finite sum-of-separable family.

Kernel special cases do not imply identical inference

(GS-)LVMOGP

$$\mathbf{Z}_H \times \mathbf{Z}_X$$

Inducing points placed separately in input and latent spaces

T-LVMOGP

$$\mathbf{Z} \in \mathbb{R}^{M \times D_T}$$

Inducing points placed directly in the learned embedding space

The special-case statement concerns the covariance class; the sparse variational approximations differ.

Variational Inference

We introduce prior and variational distributions for the latent variable $\mathbf{H}$ across outputs:

$$p(\mathbf{H}) = \prod_{p=1}^{P} p(\mathbf{h}_p) = \prod_{p=1}^{P} \mathcal{N}(\mathbf{h}_p \mid \mathbf{0}, \mathbf{I}), \qquad q(\mathbf{H}) = \prod_{p=1}^{P} q(\mathbf{h}_p) = \prod_{p=1}^{P} \mathcal{N}(\mathbf{h}_p \mid \mathbf{m}_p, \mathbf{S}_p).$$

The variational distribution of inducing variables $\mathbf{u}$ is: $q(\mathbf{u}) = \mathcal{N}(\mathbf{u} \mid \mathbf{m_u}, \mathbf{S_u})$.

Given input $\mathbf{x}_n$ and a sample of the latent variable $\mathbf{h}_p$, the embedding is computed as $\tilde{\mathbf{x}}_{n,p} = \Phi_\theta(\mathbf{x}_n, \mathbf{h}_p)$.

The variational distribution for $f_p(\mathbf{x}_n)$ given $\mathbf{h}_p$ can be computed as

$$q(f_p(\mathbf{x}_n) \mid \mathbf{x}_n, \mathbf{h}_p) = \int q(\mathbf{u}) p(f_p(\mathbf{x}_n) \mid \mathbf{h}_p, \mathbf{u}) d\mathbf{u} = \mathcal{N}(f_p(\mathbf{x}_n) \mid \mu_{n,p}, \sigma^2_{n,p}).$$

$$\mu_{n,p} = k_{\text{base}}(\tilde{\mathbf{x}}_{n,p}, \mathbf{Z}) k_{\text{base}}(\mathbf{Z}, \mathbf{Z})^{-1} \mathbf{m_u}$$

$$\begin{aligned}\sigma^2_{n,p} = {} & k_{\text{base}}(\tilde{\mathbf{x}}_{n,p}, \tilde{\mathbf{x}}_{n,p}) - k_{\text{base}}(\tilde{\mathbf{x}}_{n,p}, \mathbf{Z}) k_{\text{base}}(\mathbf{Z}, \mathbf{Z})^{-1} k_{\text{base}}(\mathbf{Z}, \tilde{\mathbf{x}}_{n,p}) \\ & + k_{\text{base}}(\tilde{\mathbf{x}}_{n,p}, \mathbf{Z}) k_{\text{base}}(\mathbf{Z}, \mathbf{Z})^{-1} \mathbf{S_u} k_{\text{base}}(\mathbf{Z}, \mathbf{Z})^{-1} k_{\text{base}}(\mathbf{Z}, \tilde{x}_{n,p})\end{aligned}$$

The MOGP problem is reformulated as a scalar GP over learned representations.

ELBO of T-LVMOGP with Stochastic Optimisation

Same as GS-LVMOGP:

$$\mathcal{L} = \underbrace{\mathbb{E}_{q(\mathbf{f}|\mathbf{H})q(\mathbf{H})}\left[\log p(\mathbf{Y} \mid \mathbf{f})\right]}_{\mathcal{T}_1} - \underbrace{\mathrm{KL}[q(\mathbf{u}) \,||\, p(\mathbf{u})]}_{\mathcal{T}_2} - \underbrace{\mathrm{KL}[q(\mathbf{H}) \,||\, p(\mathbf{H})]}_{\mathcal{T}_3}.$$

$$\mathcal{T}_1 = \sum_{p=1}^{P}\sum_{n=1}^{N} \mathbb{E}_{q(f_{n,p}|\mathbf{h}_p)q(\mathbf{h}_p)}[\log p(y_{n,p} \mid f_{n,p})]$$

$$\approx \frac{NP}{N_b P_b} \sum_{p\in\mathcal{B}_P}\sum_{n\in\mathcal{B}_N} \mathbb{E}_{q(f_{n,p}|\mathbf{h}_p)q(\mathbf{h}_p)}[\log p(y_{n,p} \mid f_{n,p})]$$

$$\approx \frac{NP}{N_b P_b} \sum_{p\in\mathcal{B}_P}\sum_{n\in\mathcal{B}_N} \frac{1}{J}\sum_{j=1}^{J} \mathbb{E}_{q(f_{n,p}|\mathbf{h}_p^{(j)})}[\log p(y_{n,p} \mid f_{n,p})]$$

where $\mathbf{h}_p^{(j)}, j \in \{1, 2, ..., J\}$ are samples from $q(\mathbf{h}_p)$.

$$\mathcal{T}_3 = \sum_{p=1}^{P} \mathrm{KL}[q(\mathbf{h}_p) \,||\, p(\mathbf{h}_p)]$$

$$\approx \frac{P}{P_b} \sum_{p\in\mathcal{B}_P} \mathrm{KL}[q(\mathbf{h}_p) \,||\, p(\mathbf{h}_p)]$$

Similar to the ELBO of the LVMOGP model.

Predictive Posterior

For a new input $\mathbf{x}_*$ and a target output dimension $p_* \in \{1, 2, ..., P\}$, the predictive distribution of $f_{p_*}(\mathbf{x}_*)$ is:

$$q(f_{p_*}(\mathbf{x}_*)) = \int q(\mathbf{h}_*) q(f_{p_*}(\mathbf{x}_*) \mid \mathbf{x}_*, \mathbf{h}_{p_*}) d\mathbf{h}_{p_*}.$$

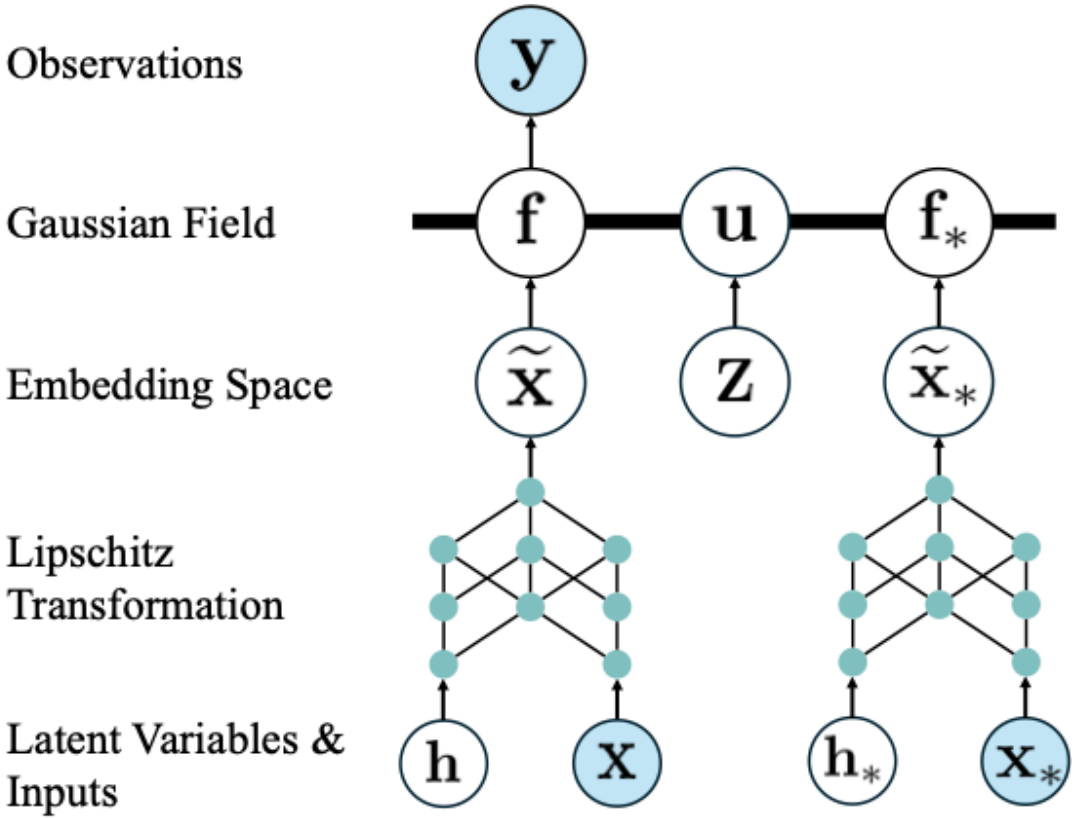
Since this integration is analytically intractable, we employ a Monte Carlo approximation. They draw S samples from $q(\mathbf{h}_{p_*})$, and obtain a mixture of Gaussians:

$$q(f_{p_*}(\mathbf{x}_*)) \approx \frac{1}{S} \sum_{s=1}^{S} \mathcal{N}(f_{p_*}(\mathbf{x}_*) \mid \mu_*^{(s)}, \sigma_*^{2(s)}).$$

The computation of $\mu_*^{(s)}$ and $\sigma_*^{2(s)}$ follows standard SVGP prediction.

If we are interested in predictive distribution of observation $\tilde{y}_*$, we further integrate $q(f_{p_*}(\mathbf{x}_*))$ through the likelihood.

Overall Structure of the T-LVMOGP model

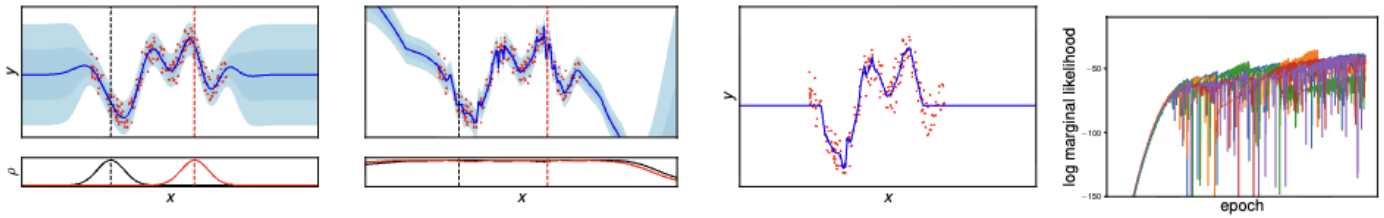


- A generalisation of (GS)-LVMOGP.
- Inducing points are placed in the embedding space.
- Mini-batch training over both inputs and outputs.
- Naturally supports non-Gaussian likelihoods.

Why Lipschitz neural network? To alleviate the "overfitting" issue of deep kernel GPs.

Issues of Deep Kernel GPs

(1) Deep kernel GPs can **over-correlate** the datapoints, which leads to **overconfident predictions** for unseen inputs.

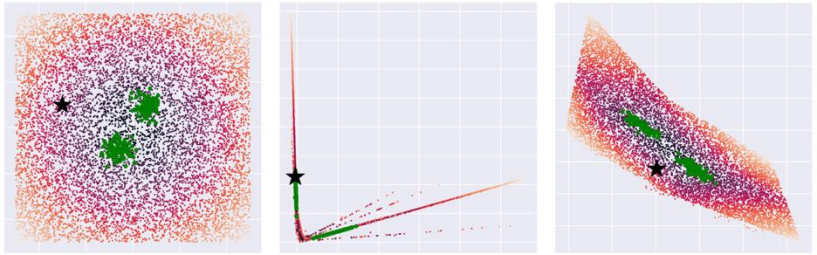


(a) SE kernel (b) Exact DKL kernel (c) Neural network fit (d) LML training curves

Figure 1: Results on toy 1D dataset. Plots (a) and (b) show the predictive posterior for squared exponential (SE) and deep kernel learning (DKL) kernels, respectively; below each plot we also plot correlation functions $\rho_{x'}(x) = k(x, x')/\sigma_f^2$ at two points x' given by the vertical dashed lines. (c) shows the fit given by the neural network analogous to the DKL model. Finally, (d) shows training curves of the log marginal likelihood (LML) for 5 different initializations of DKL.

Credit: Ober et al.

(2) Deep kernel GPs can map in- and out-of-distribution data to the **same location** in feature space: **feature collapse**. The GP will incorrectly assign high confidence to these out-of-distribution inputs.



(a) Input (b) Trained without constraint (c) Trained with constraint

(a) a 2D classification task where the classes are two Gaussian blobs (drawn in green), and a grid of unrelated points (colored according to their log-probability under the data generating distribution).
(b) the features computed by a model without constraint.
(c) the features computed by a model with Lipschitz constraint.

Credit: Van Amersfoort et al.

Deep kernel flexibility needs regularisation that preserves meaningful distances.

Ober, S. W., Rasmussen, C. E., and van der Wilk, M. The promises and pitfalls of deep kernel learning. **UAI, 2021**.
Van Amersfoort, J., Smith, L., Jesson, A., Key, O., and Gal, Y. On feature collapse and deep kernel learning for single forward pass uncertainty. arXiv preprint arXiv:2102.11409, **2021**.

Desired property: a bi-Lipschitz transformation

For some $0 < c_{\text{lower}} \leq c_{\text{upper}} < \infty$, denote $\mathbf{v} = [\mathbf{x}; \mathbf{h}]$

$$c_{\text{lower}}||\mathbf{v} - \mathbf{v}'|| \leq ||\Phi_\theta(\mathbf{v}) - \Phi_\theta(\mathbf{v}')|| \leq c_{\text{upper}}||\mathbf{v} - \mathbf{v}'||$$

lower factor c_{lower}

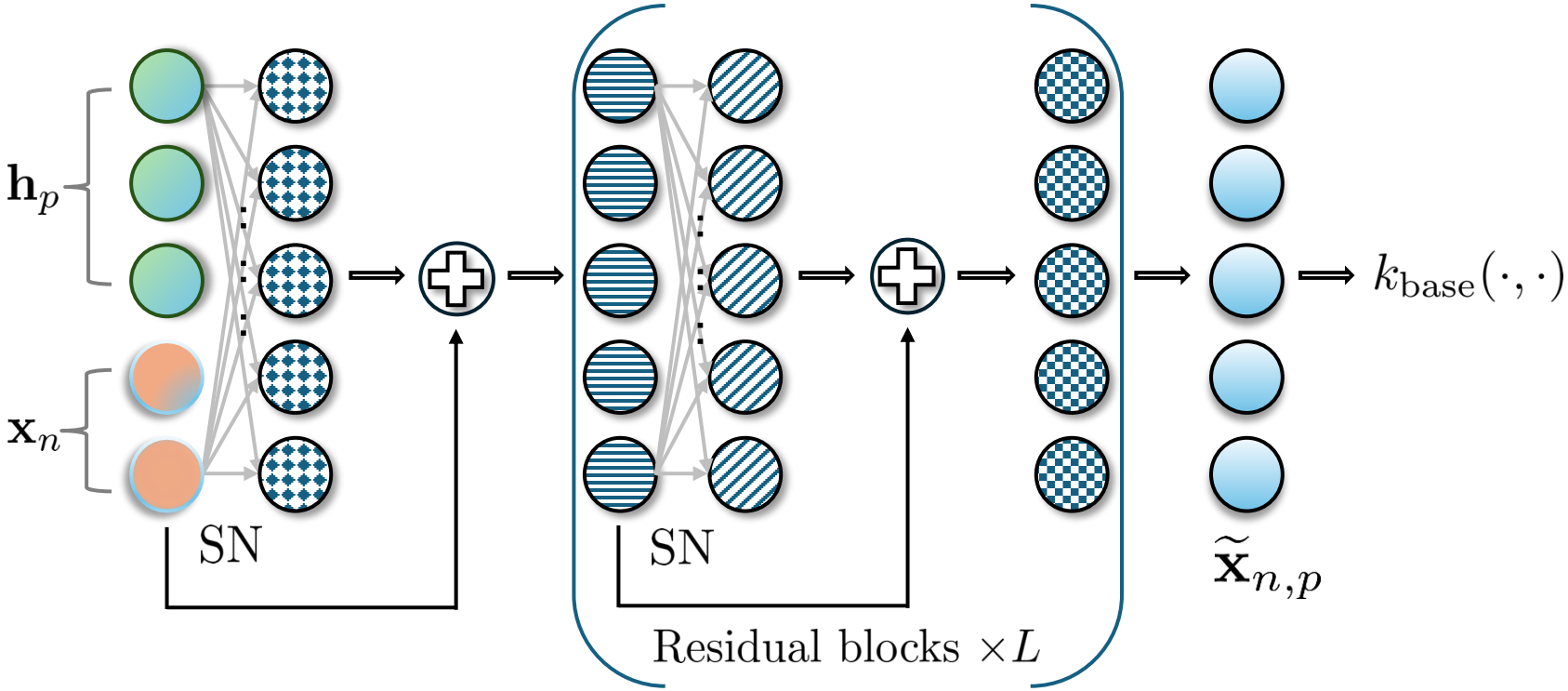
prevents arbitrary contraction
→ protects separation

Upper factor c_{upper}

prevents arbitrary expansion
→ smooth, stable mapping

How to approximately achieve this property? One approach is to use residual network with spectral normalisation.

Overview of the Neural Network



Why this structure approximately lead to bi-Lipschitz property?

bi-Lipschitz Property of Residual Connected Neural Network

For residual connected neural networks with **L blocks,** with $0 \leq \alpha < 1$

$$\Phi_\theta = \phi_L \circ \phi_{L-1} \ldots \circ \phi_1 \text{ where } \phi_l(\mathbf{v}) = \mathbf{v} + h_l(\mathbf{v}),\ ||h_l(\mathbf{v}) - h_l(\mathbf{v}')|| \leq \alpha ||\mathbf{v} - \mathbf{v}'||$$

$$(1-\alpha)^L ||\mathbf{v} - \mathbf{v}'|| \leq ||\Phi_\theta(\mathbf{v}) - \Phi_\theta(\mathbf{v}')|| \leq (1+\alpha)^L ||\mathbf{v} - \mathbf{v}'||$$

Lower factor $c_{\text{lower}} = (1-\alpha)^L$

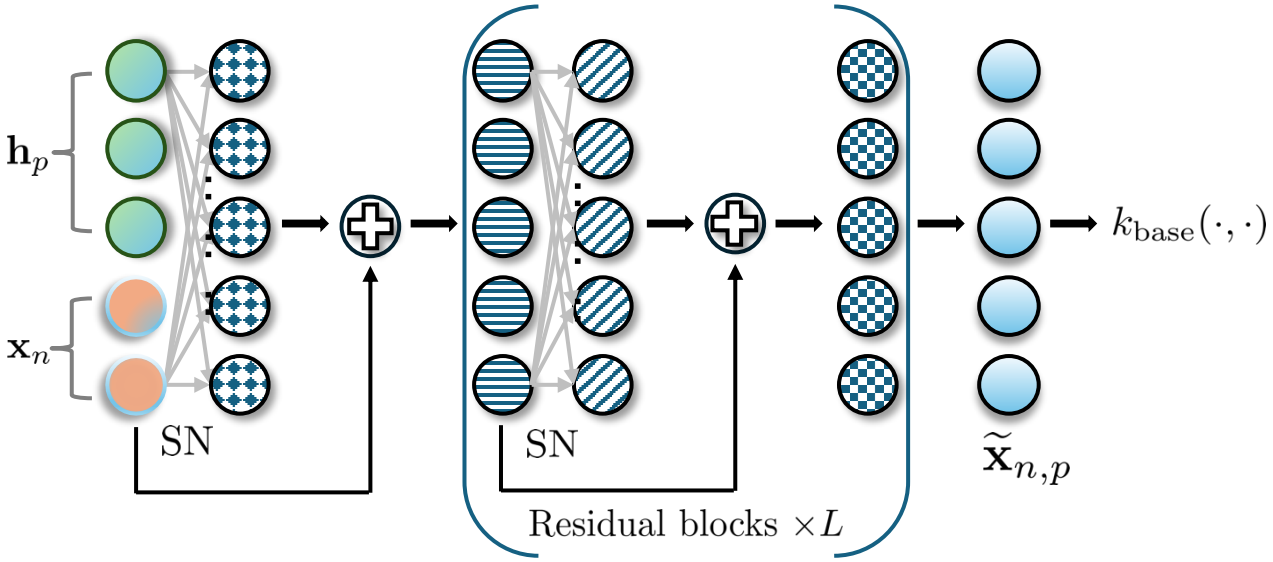
prevents arbitrary contraction
→ protects separation

Upper factor $c_{\text{upper}} = (1+\alpha)^L$

prevents arbitrary expansion
→ smooth, stable mapping

The key now becomes to "approximately ensure" α-Lipschitz **for** $h_l(\mathbf{v})$ **.**

Spectral Normalisation



We adopt $h_l(\mathbf{v}) = \text{ReLU}(\mathbf{A}_l\mathbf{v} + \mathbf{b}_l)$, the Lipschitz constant is at most $||\mathbf{A}_l||_2$, the **spectral norm** of the weight matrix.

By setting Spectral Norm Upper Bound (**SN-UB**) of $\mathbf{A}_l$ equal to α, we achieve α-Lipschitz of $h_l(\mathbf{v})$.

Definition of **spectral norm**:

$$||\mathbf{A}||_2 = \sup\{||\mathbf{A}\mathbf{v}||_2 : \mathbf{v} \in \mathbb{R}^n \text{ such that } ||\mathbf{v}||_2 \leq 1\}.$$

Spectral norm of a matrix $\mathbf{A}$ is the **largest singular value** of $\mathbf{A}$, i.e., the square root of the largest eigenvalue of the matrix $\mathbf{A}^\top\mathbf{A}$.

$$\|\mathbf{A}\|_2 = \sqrt{\lambda_{\max}\left(\mathbf{A}^\top\mathbf{A}\right)} = \sigma_{\max}(\mathbf{A}).$$

Perform spectral normalization for $\mathbf{A}$ using **Power Iteration**.

Compare LMC, LVMOGP, GS-LVMOGP and T-LVMOGP

Model	Output representation	Cross-output covariance	Structure	Large P
LMC	rows of $\mathbf{A}$.	Linear kernel	Sum of separable	Possible, via SV-LMC
LVMOGP	latent variables $\mathbf{H}$.	Any valid kernel	Single separable	Limited
GS-LVMOGP	Q groups of latent variables $\mathbf{H} = \{\mathbf{H}^{(q)}\}_{q=1}^{Q}$.	Any valid kernel	Sum of separable	Yes*
T-LVMOGP	**latent variables $\mathbf{H}$.**	**Any valid base kernel, neural network $\Phi_\theta(\cdot)$.**	**Implicit**	**Yes***

*The computational cost of each iteration does not increase with P.

Computational Complexity

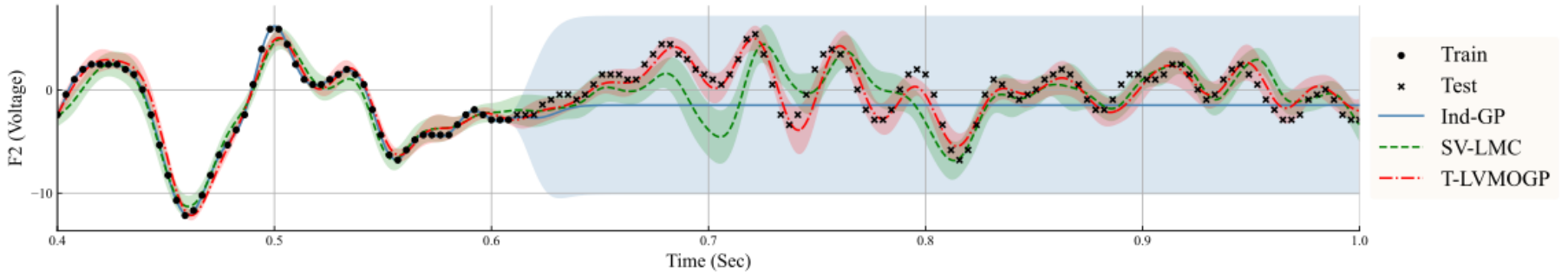
The computational complexity mainly stems from **three** components:

- The SVGP computation, which scales as $\mathcal{O}(N_b P_b M^2 + M^3)$.
- The spectral normalisation operations: enforcing Lipschitz constraints via power iteration incurs $\mathcal{O}(Tmn)$ for a $m \times n$ matrix over T iterations.
- Forward and backward propagation of the neural network, which is $\mathcal{O}(N_b P_b mn)$ for a linear layer with a $m \times n$ weight matrix.

For relatively small width and depth of the neural networks used in our experiments, the SVGP computation dominates the overall computational overhead.

EXPERIMENTS: All results are reported as mean ± standard deviation over 10 random trials.

Small-P tasks: EEG (N=256; P=7)



Ind-GP: Independent GP
SGPRN [Li et al., 2020]
OILMM [Bruinsma et al., 2020]
SV-LMC [Van der Wilk et al., 2020]
G-MOGP [Dai et al., 2024]
GS-LVMOGP [Jiang et al., 2025]

Table 1. Performance of different models on the EEG dataset. The ↓ symbol indicates that lower values are better.

Model	MSE ↓	NLL ↓
Baselines		
Ind-GP	0.466 ± 0.000	1.227 ± 0.000
SGPRN	0.473 ± 0.003	7.105 ± 2.110
G-MOGP	0.604 ± 0.099	1.337 ± 0.038
OILMM	0.372 ± 0.098	0.979 ± 0.159
GS-LVMOGP	0.366 ± 0.064	0.924 ± 0.081
SV-LMC	0.282 ± 0.193	0.857 ± 0.613
T-LVMOGP (ours)	$\mathbf{0.115 \pm 0.025}$	$\mathbf{0.814 \pm 0.310}$

Van der Wilk, M., Dutordoir, V., John, S., Artemev, A., Adam, V., and Hensman, J. A framework for interdomain and multioutput Gaussian processes. arXiv preprint arXiv:2003.01115, **2020**.
Bruinsma, W., Perim, E., Tebbutt, W., Hosking, S., Solin, A., and Turner, R., Scalable exact inference in multi-output Gaussian processes. **ICML, 2020**.
Li, S., Xing, W., Kirby, R. M., and Zhe, S., Scalable Gaussian process regression networks. **IJCAI, 2020**.
Dai, Y., Yan, W., and Yin, F. Graphical multioutput Gaussian process with attention. **ICLR, 2024**.
Jiang, X., Georgaka, S., Rattray, M., and Álvarez, M., Scalable multi-output Gaussian processes with stochastic variational inference. **TMLR, 2025**.

Small-P tasks: SARCOS (N=48,933; P=7)

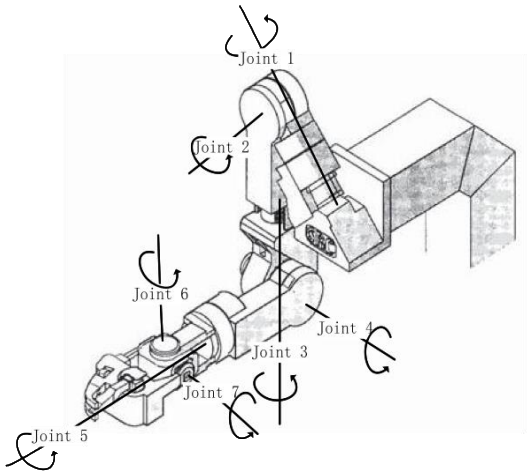
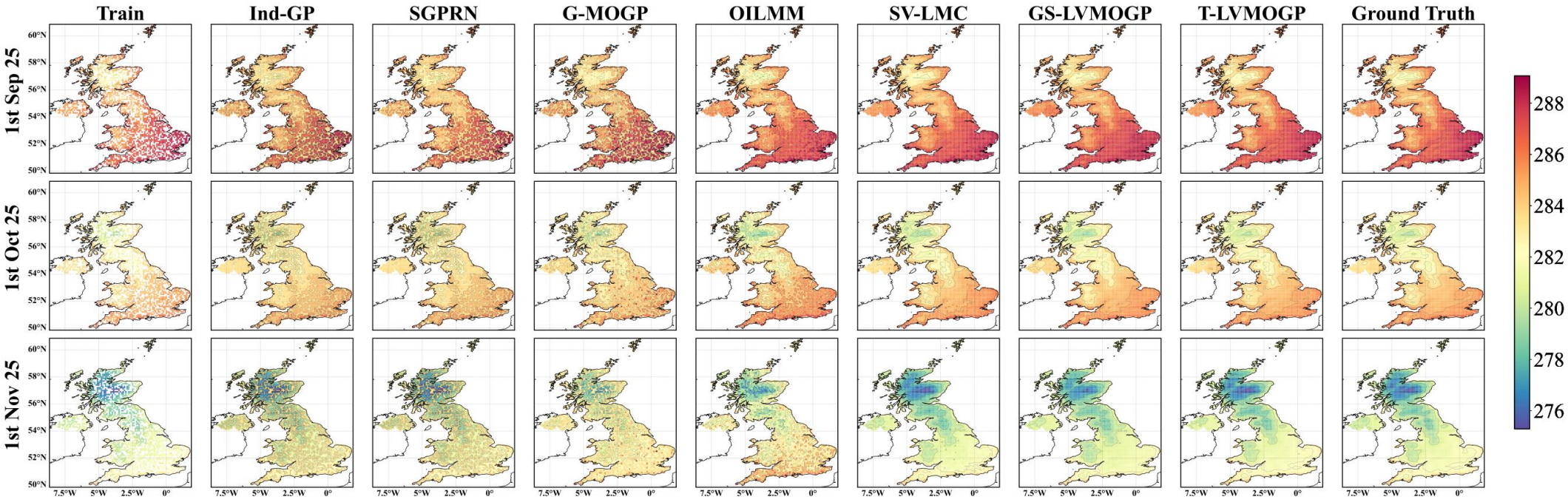


Table 3. Performance comparison of different models on the SARCOS dataset.

Model	MSE ↓	NLL ↓	Training Time (s/epoch) ↓
Baselines			
OILMM (Bruinsma et al., 2020)	0.140 ± 0.010	0.864 ± 0.090	8.951 ± 0.070
GS-LVMOGP (Jiang et al., 2025)	0.037 ± 0.001	-0.220 ± 0.011	10.241 ± 0.096
SV-LMC (Van der Wilk et al., 2020)	0.033 ± 0.000	-0.297 ± 0.003	6.185 ± 0.086
G-MOGP (Dai et al., 2024)	0.023 ± 0.001	-0.483 ± 0.017	5.892 ± 0.084
T-LVMOGP (ours)	$\mathbf{0.022 \pm 0.000}$	$\mathbf{-0.485 \pm 0.009}$	$\mathbf{5.263 \pm 0.194}$

T-LVMOGP matches the best predictive accuracy while achieving the lowest reported training time.

ERA5: Block-wise Spatiotemporal Imputation (N=30; P=3,395)



T-LVMOGP

MSE:0.003 ± 0.000

NLL:-1.503 ± 0.026

best baseline: GS-LVMOGP

MSE:0.019 ± 0.011

NLL:-0.550 ± 0.230

T-LVMOGP reduces ERA5 block-split MSE from 0.019 to 0.003 versus the best baseline.

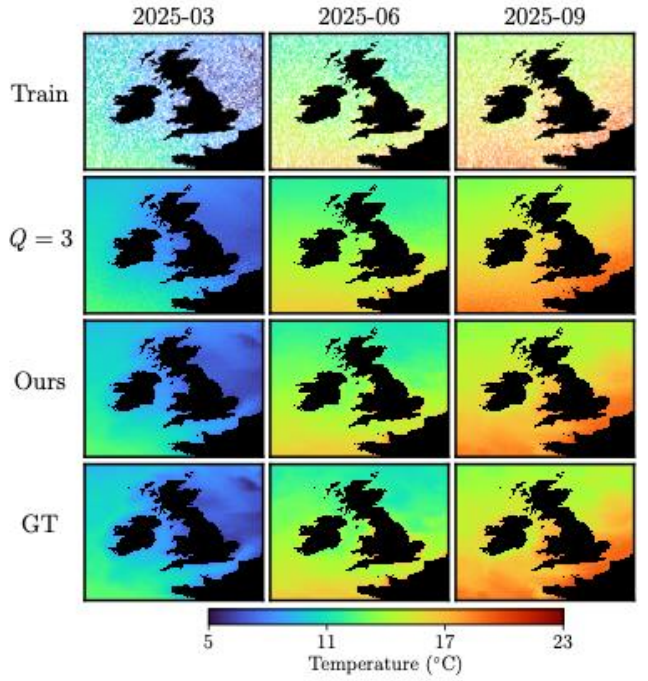
Copernicus Marine dataset: Output Extrapolation (N=24; P=10,873)

We consider the **extrapolation-of-outputs** setting:

- Spatial coordinates serve as the prior mean for the latent variables.
- The model is trained on a subset of spatial locations to generate predictions for the entire 24-month period at new locations.
- This dataset contains 21, 679 outputs, with 10, 873 observed and 10, 806 held out for test.

Table 6. Performance comparison on the Copernicus Marine dataset for the output extrapolation task.

	GS-LVMOGP (Jiang et al., 2025)			**T-LVMOGP** (ours)
Metric	Q=1	Q=2	Q=3	RCNN
MSE ↓	0.040 ± 0.002	0.036 ± 0.001	0.035 ± 0.002	$\mathbf{0.029 \pm 0.000}$
NLL ↓	5.230 ± 0.640	5.086 ± 0.297	4.975 ± 0.923	$\mathbf{-0.439 \pm 0.011}$
Training Time (s/epoch) ↓	1.321 ± 0.043	1.750 ± 0.045	2.076 ± 0.048	$\mathbf{1.230 \pm 0.012}$



Predictions of GS-LVMOGP with Q=3 and T-LVMOGP for output extrapolation task.

Follows the same setting as in [Jiang et al., 2025].

Spatial Transcriptomics (N=4,352; P=5,000)

10x Genomics Visium human prostate cancer dataset.

- Integer value observations.
- High sparsity and significant over-dispersion.
- We employ the Zero-Inflated Negative Binomial (ZINB) likelihood.

Table 7. Predictive performance of GS-LVMOGP and T-LVMOGP on a Spatial Transcriptomics dataset.

	GS-LVMOGP (Jiang et al., 2025)			**T-LVMOGP** (ours)
Metric	Q=1	Q=2	Q=3	RCNN
MSE ↓	11.616 ± 0.773	11.396 ± 0.314	11.024 ± 0.350	$\mathbf{9.189 \pm 0.342}$
NLL ↓	0.677 ± 0.002	0.675 ± 0.001	$\mathbf{0.674 \pm 0.001}$	$\mathbf{0.674 \pm 0.001}$
Training Time (s/epoch) ↓	$\mathbf{65.507 \pm 0.820}$	73.639 ± 0.505	78.245 ± 1.131	73.047 ± 0.318

MSE improves, NLL matches GS-LVMOGP $Q = 3$, runtime is comparable to $Q = 2$.

Ablation Study

Spectral Normalisation

Table 8. Test NLL comparison of T-LVMOGP with and without spectral normalisation.

Dataset	w/o SN	w/ SN
EEG	4.109 ± 2.141	$\mathbf{0.814 \pm 0.310}$
SARCOS	0.112 ± 0.856	$\mathbf{-0.485 \pm 0.01}$
ERA5 (random)	-1.401 ± 0.13	$\mathbf{-1.564 \pm 0.03}$
ERA5 (block-wise)	-0.895 ± 0.85	$\mathbf{-1.503 \pm 0.03}$
Copernicus Marine	-0.400 ± 0.08	$\mathbf{-0.439 \pm 0.01}$

Neural Network

Table 9. Test NLL comparison of T-LVMOGP with and without neural network architecture.

Dataset	w/o NN	w/ NN
EEG	1.153 ± 0.437	$\mathbf{0.814 \pm 0.310}$
SARCOS	-0.336 ± 0.01	$\mathbf{-0.485 \pm 0.01}$
ERA5 (random)	-1.554 ± 0.03	$\mathbf{-1.564 \pm 0.03}$
ERA5 (block-wise)	-1.474 ± 0.06	$\mathbf{-1.503 \pm 0.03}$
Copernicus Marine	-0.420 ± 0.01	$\mathbf{-0.439 \pm 0.01}$

Tighter Variational Bounds

Table 10. Test NLL comparison of T-LVMOGP with standard and tighter variational bounds. †: for non-Gaussian likelihood, tighter bounds might not be effectively tighter in practice; more explanations are provided in Appendix F.7.

Dataset	Standard	Tighter
SARCOS	-0.485 ± 0.01	$\mathbf{-0.502 \pm 0.01}$
Copernicus Marine	-0.439 ± 0.01	$\mathbf{-0.443 \pm 0.01}$
Spatial Transcriptomics†	0.674 ± 0.001	0.674 ± 0.002

T-LVMOGP

Why MOGP?

Share information across outputs instead of fitting P isolated functions.

Why latent variables?

Generalise LMC's linear output kernel to a flexible kernel over latent variables.

Why transform?

Learn complex, non-separable cross-output relationships.

Why Lipschitz?

To retain distance awareness of the learned transformation and to improve the uncertainty of deep kernel GPs.

Future work structured q(H) · amortised output inference · SKI / nearest-neighbour GP as scalability backbones

Code: github.com/XiaoyuJiang17/T-LVMOGP-official

Questions?