Efficient Transformed Gaussian Processes for Non-Stationary Dependent Multi-class Classification

Juan Maroñas [1 2] **Daniel Hernández-Lobato** [1]

Abstract

This work introduces the Efficient Transformed Gaussian Process (ETGP), a new way of creating C stochastic processes characterized by: 1) the C processes are *non-stationary*, 2) the C processes are dependent by construction without needing a mixing matrix, 3) training and making predictions is very efficient since the number of Gaussian Processes (GP) operations (e.g. inverting the inducing point's covariance matrix) do not depend on the number of processes. This makes the ETGP particularly suited for multi-class problems with a very large number of classes, which are the problems studied in this work. ETGP exploits the recently proposed Transformed Gaussian Process (TGP), a stochastic process specified by transforming a Gaussian Process using an invertible transformation. However, unlike TGP, ETGP is constructed by transforming a single sample from a GP using C invertible transformations. We derive an efficient sparse variational inference algorithm for the proposed model and demonstrate its utility in 5 classification tasks which include low/medium/large datasets and a different number of classes, ranging from just a few to hundreds. Our results show that ETGP, in general, outperforms state-of-the-art methods for multi-class classification based on GPs, and has a lower computational cost (around one order of magnitude smaller).

1. Introduction

Gaussian Processes (GPs) are stochastic processes characterized by their finite-dimensional distributions being multivariate Gaussian (Rasmussen & Williams, 2006), which have become a uniquely popular modeling tool. For example, in the machine learning community GPs are used as prior distributions over functions, used to solve tasks such as regression, classification, feature extraction or hyperparameter optimization (Rasmussen & Williams, 2006; Lawrence, 2003; Snoek et al., 2012). Their non-parametric nature means that they become more expressive with more data (Rasmussen & Williams, 2006). Furthermore, GPs are characterized by a predictive distribution which provides information about what the model does not know (Gal, 2016) and are easy to interpret since the covariance function gives insights about the nature of the latent function to be inferred (Duvenaud et al., 2013). GPs have also been applied in spatial statistics (Krige, 1951), and to explain physics phenomena such as those that arise when studying molecular dynamics (Leimkuhler & Matthews, 2015; Uhlenbeck & Ornstein, 1930). Moreover, they are used as a theoretical tool to understand Deep Neural Networks (DNN) (Neal, 1996; Yang, 2019) and lie at the core of a recent family of Deep Generative Models that generate samples attending to the dynamics of a diffusion process (Song et al., 2021).

In this paper, we focus on multi-class classification problems with $C > 2$ classes. For this, one often defines C independent GPs, one per each class (Rasmussen & Williams, 2006). In this case, the number of GP operations (such as inverting the covariance matrix associated to the inducing points) grows linearly with C. Thus, if C is large, this can be too expensive. Some speed-up tricks include sharing the inducing points or the covariance function across GPs, but this may reduce prediction performance, as we show in our experiments. Even with this trick, computing the predictive distribution still has complexity $\mathcal{O}(CM^2)$ per datapoint.

We can gain additional performance by defining a prior process using C dependent GPs, which can be done by combining Q latent GPs with a mixing matrix $\mathbf{\Phi} \in \mathbb{R}^{Q \times C}$. In practice, however, these dependencies are often ignored since the memory complexity scales as $\mathcal{O}(C^2)$ per datapoint. In fact, modern SOTA GP software's like GPFLOW (Matthews et al., 2017; van der Wilk et al., 2020) require significant source code modification (up to early 2023) to handle these dependencies in an efficient way.

A disadvantage of GPs is that they usually impose strong

[1]Machine Learning Group, Universidad Autónoma de Madrid, Madrid, Spain [2]Work done previous to joining Cognizant, Madrid, Spain. Correspondence to: Juan Maroñas <juan.maronnas@uam.es, jmaronasm@gmail.com>.

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

assumptions about the nature of the latent function. For example, most covariance functions are stationary and assume a constant level of smoothness for the latent function on the input domain (Rasmussen & Williams, 2006). If this is not the case, the performance can be degraded. GPs can be made more expressive using non-stationary processes. However, this is usually only justified if one has prior knowledge about the nonstationarity of the particular application, e.g. in Bayesian Optimization (Snoek et al., 2014), Geostatistics (Wang et al., 2020; Wilson et al., 2012; Hamelijnck et al., 2019; Sampson & Guttorp, 1992; Schmidt et al., 2000) or temporal gene expression modeling (Heinonen et al., 2016).

The flexibility of GPs can also be increased by using non-linear transformations. Examples include deep GPs (DGPs) (Damianou & Lawrence, 2013) and transformed GPs (TGPs) (Maroñas et al., 2021). In DGPs, the output of a GP is used as the input of another GP systematically, following a fully connected neural network (NN) architecture in which units are GPs. As a result of the concatenation, the resulting process need not be stationary. By contrast, in TGPs the initial GP prior is transformed iteratively using input-dependent invertible transformations (Maroñas et al., 2021). Because of this input dependence, the resulting process need not be stationary. Importantly, TGPs often generate models that are as accurate as DGPs at a lower computational cost.

In this work, we introduce the Efficient Transformed Gaussian Process (ETGP), a new model where C processes are specified by sampling from a single GP, and then transforming this sample using C invertible transformations (throughout the paper we refer to the invertible transformations by flows or warping functions as well). By construction, the C processes are non-stationary (due to the input-dependence of the transformation) and dependent, with dependencies modeled by the copula of the base GP[1]. Importantly, ETGP does not have the computational and memory complexity of an equivalent number of GPs, since only one GP is used in the construction of the C processes. A special case of the ETGP family specified by using a linear flow includes non-stationary dependent GPs, which we also characterize and study. We derive an efficient sparse variational inference (VI) algorithm for training ETGPs and evaluate its prediction performance and computational cost in the context of multi-class problems with a large number of classes C. More precisely, we carry out experiments on several large and small datasets with up to 153 class labels. The results obtained show that ETGP, in general, outperforms SOTA methods for multi-class classification based on GPs. Moreover, ETGP has a computational cost that is around one order of magnitude smaller. Finally, our experiments also show that usual non-stationary covariance functions are not useful for black-box function approximation and that the particular inductive bias of the ETGP gives better results.

[1]This might not be the case for non-diagonal transformations (Maroñas et al., 2021; Rios, 2020), not considered in this work.

2. Background

We start by introducing GPs for multi-class classification problems and some notation. We also describe how to improve GPs using the TGP method of Maroñas et al. (2021).

2.1. Multi-class Gaussian Process Classification

Consider assigning a class label $y \in \mathcal{Y} = \{1, \dots, C\}$, with C the number of classes, to an input $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$. Our goal is to learn C functions mapping $\mathbf{x}$ to class label probabilities. For this, we are given a set of N instances $\mathcal{D} = \{\mathbf{x}^n, y^n\}_{n=1}^N$ generated from some distribution. Define $\mathbf{X} = (\mathbf{x}^1, \dots, \mathbf{x}^N)$ and $\mathbf{y} = (y^1, \dots, y^N)$. One can model each function by placing an independent GP over each of them. This prior is then updated into a posterior over functions given $\mathcal{D}$ using Bayes rule, resulting in a predictive distribution for the label of new data.

A GP is a stochastic process whose finite-dimensional distributions are given by a multivariate Gaussian. Specifically, let $\mathbf{f} = (f(\mathbf{x}^1), \dots, f(\mathbf{x}^N))^{\mathrm{T}}$ and define $f_n \coloneqq f(\mathbf{x}^n)$. Then, $\mathbf{f} \sim \mathcal{N}(\mu_\nu(\mathbf{X}), K_\nu(\mathbf{X}, \mathbf{X}))$, where the mean vector $\mu_\nu(\mathbf{X}) = (\mu_\nu(\mathbf{x}^1), \dots, \mu_\nu(\mathbf{x}^N))^{\mathrm{T}}$ is given by a mean function $\mu_\nu : \mathcal{X} \to \mathbb{R}$, and $K_\nu(\mathbf{x}, \mathbf{x})$ is a $N \times N$ matrix whose i-th row and j-th column entry is $K_\nu(\mathbf{x}^i, \mathbf{x}^j)$, given by $K_\nu : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, a covariance function; both parameterized by ν. However, without loss of generality, we assume zero mean GPs.

Consider C independent GPs and denote $\overline{\mathbf{f}} = \{\mathbf{f}^1, \dots \mathbf{f}^C\}$, where a bar over a letter $\overline{x}$ summarizes the corresponding C elements $x^1, \dots, x^C$. Often, a Softmax link function $\pi_c(\overline{f}_n) = \exp(f^c(\mathbf{x}^n)/\sum_{c'=1}^C \exp(f^{c'}(\mathbf{x}^n))$ is applied to $\overline{\mathbf{f}}$ to obtain class label probabilities π_c (Rasmussen & Williams, 2006). Then, these probabilities are linked to actual class labels $\mathbf{y}$ by a categorical likelihood. Under these conditions, the joint distribution of $\mathbf{y}$ and $\overline{\mathbf{f}}$ is:

$$p(\mathbf{y}, \overline{\mathbf{f}}) = p(\mathbf{y} \mid \overline{\mathbf{f}}) p(\overline{\mathbf{f}}) = \left[\prod_{n=1}^N \prod_{c=1}^C \pi_c(\overline{f}_n)^{\mathbb{I}(y^n=c)}\right] \times \prod_{c=1}^C \mathcal{N}(\mathbf{f}^c \mid \mathbf{0}, K_\nu^c(\mathbf{X}, \mathbf{X})), \quad (1)$$

where $\mathbb{I}(\cdot)$ is the indicator function. To make predictions we need to approximate the intractable posterior $p(\overline{\mathbf{f}} \mid \mathcal{D})$. However, even if tractable, its computation would scale cubically with N (Rasmussen & Williams, 2006). For both reasons, we use approximate sparse variational inference (VI) GPs (SVGPs) (Chai, 2012; Titsias, 2009), with the modifications introduced in Hensman et al. (2013) to scale to very large datasets. SVGPs work by introducing a set of $M \ll N$ inducing points locations $\mathbf{Z} = (\mathbf{z}^1, \dots, \mathbf{z}^M), \mathbf{z} \in \mathcal{X}$ with associated process values $\mathbf{u} = (f(\mathbf{z}^1), \dots, f(\mathbf{z}^M))^{\mathrm{T}}$ per

each of the C functions (Titsias, 2009). The joint Gaussian prior $p(\overline{\mathbf{f}}, \overline{\mathbf{u}} \mid \mathbf{X}, \overline{\mathbf{Z}})$ is easily obtained in terms of the covariance function. These inducing points act as sufficient statistics of the data $\mathbf{x}$, and represent the posterior $p(\overline{\mathbf{f}} \mid \mathcal{D})$ compactly using M points, reducing the overall training complexity from $\mathcal{O}(CN^3)$ to $\mathcal{O}(CM^2)$.

The key point of Titsias (2009) is to treat $\overline{\mathbf{Z}}$ as variational parameters, which are optimized by minimizing the Kullback-Leibler Divergence (KL) between a variational posterior $q(\overline{\mathbf{f}}, \overline{\mathbf{u}})$ and the augmented joint posterior $p(\overline{\mathbf{f}}, \overline{\mathbf{u}} \mid \mathcal{D}, \overline{\mathbf{Z}})$, or equivalently by maximizing the Evidence Lower Bound (ELBO). Since $\overline{\mathbf{Z}}$ are variational parameters, they are protected from overfitting. The speed-up is achieved by constraining the form of the variational distribution $q(\overline{\mathbf{f}}, \overline{\mathbf{u}}) = \prod_{c=1}^{C} p(\mathbf{f}^c \mid \mathbf{u}^c) q(\mathbf{u}^c)$, which is defined using the conditional model's prior $p(\mathbf{f}^c \mid \mathbf{u}^c)$ and a Gaussian variational distribution $q(\mathbf{u}^c)$ with mean and covariance matrix $\mathbf{m}^c \in \mathbb{R}^M$ and $\mathbf{S}^c \in \mathbb{R}^{M \times M}$. With this, the ELBO is:

$$\begin{aligned} \text{ELBO} = &\sum_{n=1}^{N} \sum_{c=1}^{C} \mathbb{I}(y^n = c) \mathbb{E}_q \left[\log \pi_c(\overline{f}_n) \right] \\ &- \sum_{c=1}^{C} \text{KL}[q(\mathbf{u}^c) \,||\, p(\mathbf{u}^c))], \end{aligned} \tag{2}$$

where $\text{KL}[q(\mathbf{u}^c) \,||\, p(\mathbf{u}^c))]$ can be computed in closed form and the expectation with respect to q can be approximated by Monte Carlo. The above expression allows using stochastic VI to optimize the ELBO, by sub-sampling the data using mini-batches (Hensman et al., 2013). We use path-wise derivatives for black-box low-variance gradient estimations. The variational distribution $q(f^c(\mathbf{x}^n)) = \int p(f^c(\mathbf{x}^n) \mid \mathbf{u}^c) q(\mathbf{u}^c) \mathrm{d}\mathbf{u}^c$ is Gaussian with mean and covariance given by $K_\nu^c(\mathbf{x}^n, \mathbf{Z}^c) K_\nu^c(\mathbf{Z}^c, \mathbf{Z}^c)^{-1} \mathbf{m}^c$ and $K_\nu^c(\mathbf{x}^n, \mathbf{x}^n) - K_\nu^c(\mathbf{x}^n, \mathbf{Z}^c) K_\nu^c(\mathbf{Z}^c, \mathbf{Z}^c)^{-1} [K_\nu^c(\mathbf{Z}^c, \mathbf{Z}^c) + \mathbf{S}^c] K_\nu^c(\mathbf{Z}^c, \mathbf{Z}^c)^{-1} K_\nu^c(\mathbf{Z}^c, \mathbf{x}^n)$, respectively. The predictive distribution for the label $y^\star$ associated to a new point $\mathbf{x}^\star$, can also be efficiently approximated via Monte Carlo by sampling from $q(f^c(\mathbf{x}^\star))$ for $c = 1, \ldots, C$.

2.2. Transformed Gaussian Processes

A limitation of GPs is that they place strong assumptions over the latent function. This can be for example assuming the same level of smoothness across the input domain, as it is often done by considering a stationary covariance function (Rasmussen & Williams, 2006). The Transformed Gaussian Process (TGP) (Maroñas et al., 2021) is a model that increases the flexibility of GPs by non-linearly and input-dependently transforming the GP prior using invertible transformations. We describe the TGP, since is the building block of the proposed approach ETGP.

Let $f_0(\cdot) \sim \text{GP}(0, K_\nu(\cdot, \cdot))$ be a GP sample. Consider a composition of K individual invertible mappings $\mathbb{G}_{\boldsymbol{\theta}} = \mathbb{G}_{\theta_0} \circ \mathbb{G}_{\theta_1} \ldots \circ \mathbb{G}_{\theta_{K-1}} : \mathcal{F}_0 \to \mathcal{F}_K$ each parameterized by $\theta \in \Theta$, with $\boldsymbol{\theta} = \{\theta_0, \theta_1, \ldots, \theta_{K-1}\}$. The TGP, is defined by the following generative procedure:

$$f_0(\cdot) \sim \text{GP}(0, K_\nu(\cdot, \cdot)), \qquad f_K(\cdot) = \mathbb{G}_{\boldsymbol{\theta}}(f_0(\cdot)). \tag{3}$$

An easy way to specify $\mathbb{G}_{\boldsymbol{\theta}}$ so that $p(\mathbf{f}_K)$ is a consistent finite-dimensional distribution (i.e. Kolmogorov extension theorem applies) is to use element-wise mappings (also known by diagonal flows), characterized by: $\forall \mathbf{x}^n \in \mathcal{X}$, $f_K(\mathbf{x}^n) = \mathbb{G}_{\boldsymbol{\theta}}(f_0(\mathbf{x}^n))$, as described in Rios (2020)[2]. Due to these element-wise mappings, the resulting process is a Gaussian Copula process (Wilson & Ghahramani, 2010) since it has arbitrary marginals with dependencies driven by the copula of the GP, something derived from Sklar's theorem (Sklar, 1959). In other words, f_0 and f_K share the same dependencies but differ in their marginal distributions. By increasing K, we can make the flow as complicated as we want, increasing flexibility.

Maroñas et al. (2021) show that one can create non-stationary processes f_K by making the parameters of the flow depend on the input using a Neural Network (NN) $\text{NN} : \mathcal{X} \to \Theta$ parameterized by $\mathbf{W}$. In Maroñas et al. (2021) they show that this non-stationary process is way more expressive than a GP and also than a stationary TGP. Since the parameters of the NN play the same role of hyper-parameters in a GP, this work shows how using a Bayesian Neural Network (BNN), with prior distribution $p(\mathbf{W} \mid \lambda)$, can effectively avoid over-fitting as this makes the NN's parameters play the same role in the graphical model as GP's latent functions, see Fig. 1.

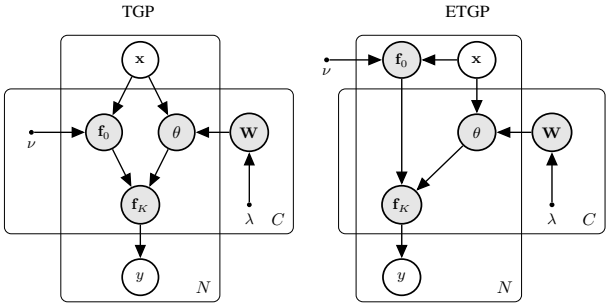

Figure 1. Prob. graphical models for TGP (left) and ETGP (right).

In a classification problem, we can write the joint conditional distribution over the C independent TGPs by applying the change of variable formula and inverse function theorem (Maroñas et al., 2021):

$$\begin{aligned} &p(\overline{\mathbf{f}}_K \mid \overline{\mathbf{W}}) = \\ &\prod_{c=1}^{C} p(\mathbf{f}_0 \mid \nu^c) \prod_{k=0}^{K^c-1} |\det \tfrac{\partial \mathbb{G}_{\theta_k}^c(\mathbf{w}^c, \mathbf{x})(\mathbf{f}_k^c)}{\partial \mathbf{f}_k^c}|^{-1}. \end{aligned} \tag{4}$$

[2]App. A gives all mathematical details about equations describing the method an inference algorithm, which assumes diagonal flows. See e.g. the appendix of Maroñas et al. (2021) for the particularities that might arise for non-diagonal flows.

This warping procedure can also be used when defining a variational approximation $q(\mathbf{f}_K, \mathbf{u}_K) = p(\mathbf{f}_K \mid \mathbf{u}_K)q(\mathbf{u}_K)$, where the cancellations of several factors in the ELBO result in an efficient training algorithm (Maroñas et al., 2021). Note that we use K^c since each flow for each class can have a different functional form (linear, SAL, etc) and a different number of compositions.

3. Efficient Transformed Gaussian Processes

In this section, we describe the proposed method for multi-class GP classification, which we show is a more efficient application of the TGP to classification problems with large C. Moreover, it has the additional benefit of naturally modeling dependencies among the latent functions. Our proposed method (ETGP) is specified by transforming a single sample from a GP using C invertible transformations, each of them mapping the GP to a latent function, one for each class label. The generative process is:

$$\begin{aligned} f_0(\cdot) &\sim \text{GP}(0, K_\nu(\cdot,\cdot)), \\ f_K^1(\cdot) &= \mathbb{G}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{x})}(f_0(\cdot)), \\ &\vdots \\ f_K^C(\cdot) &= \mathbb{G}^C_{\boldsymbol{\theta}(\mathbf{W}^C,\mathbf{x})}(f_0(\cdot)). \end{aligned} \tag{5}$$

Using again the change of variable formula and inverse function theorem employed in Sec. 2.2, the joint conditional distribution of the C processes is given by:

$$\begin{aligned} p(\overline{\mathbf{f}}_K \mid \overline{\mathbf{W}}) &= p(\mathbf{f}_0) \textstyle\prod_{k=0}^{K^1-1} |\det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1,\mathbf{x})}(\mathbf{f}_k^1)}{\partial \mathbf{f}_k^1}|^{-1} \\ &\times \textstyle\prod_{c=2}^{C} \delta(\mathbf{f}_K^c - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c,\mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1)), \end{aligned} \tag{6}$$

where $\delta(\cdot)$ denotes the Dirac measure and we define $\mathbb{H} = \mathbb{G}^{-1}$ to be the corresponding inverse transformation. Note that this decomposition is not unique. We label $\mathbf{f}_K^1$ as the *pivot* and note that this joint distribution can be written equivalently w.r.t. any other *pivot* $\mathbf{f}_K^c$ with $c \neq 1$.

Our formulation has the advantage that a single GP is used in practice. This implies a constant scaling of GP operations w.r.t. the number of classes, speeding-up computations. By contrast, a naive use of TGPs or the model described in Sec. 2.1 involves as many GPs as class labels. This motivates the name given to our method: the *Efficient* Transformed Gaussian Processes (ETGP), particularly suited for multi-class problems with large C. Moreover, the resulting C processes are dependent since they share the same latent sample f_0 from the original process. Also, since we use NNs to parameterize the flows, the C processes are non-stationary. Fig. 1 compares the graphical models of the naive use of TGP for multi-class classification and the proposed method ETGP, highlighting their differences.

We describe ETGP as a good performance method at a low computational cost. Therefore, we also propose an efficient NN parameterization of the flow parameters, illustrated in Fig. 2. In principle, we could use a NN per flow parameter, implemented efficiently using batched operations (Blackford et al., 2002). However, by using a single NN whose output layer coincides with the number of parameters per flow times the number of classes, we achieve a more efficient parameterization. For example, if we are modeling Imagenet ($C = 1000$) with a linear flow ($|\theta| = 2$), then we would use an output layer of 2000 neurons. Besides being more efficient, this could also serve as a possible regularizer in the same fashion as sharing parameters regularizes DNNs, something commonly used in computer vision applications through a back-bone convolutional model.

Another appealing property of the ETGP is that we can efficiently create C non-stationary dependent GPs, by using a linear flow: $f_K(\mathbf{x}) = a(\mathbf{x})f_0(\mathbf{x}) + b(\mathbf{x})$, as shown in Fig. 2. The next proposition characterizes the corresponding joint conditional prior distribution $p(\overline{\mathbf{f}}_K \mid \overline{\mathbf{W}})$:

Proposition 3.1. *The joint conditional distribution of C non-stationary GPs obtained via a linear flow is given by:*

$$\begin{aligned} p(\overline{\mathbf{f}}_K \mid \overline{\mathbf{W}}) &= \mathcal{N}(\mathbf{f}_K^1 \mid \mathbf{b}^1, \mathbf{A}_1 K_\nu(\mathbf{X},\mathbf{X})\mathbf{A}_1^T) \\ &\textstyle\prod_{c=2}^{C} \delta(\mathbf{f}_K^c - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c,\mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1)), \end{aligned}$$

with $\mathbf{b}^1 = (b^1(\mathbf{x}^1), \dots, b^1(\mathbf{x}^N))^T$ and $\mathbf{A}_1 \in \mathbb{R}^{N\times N}$ a diagonal matrix with entries $\mathbf{a}^1 = (a^1(\mathbf{x}^1), \dots, a^1(\mathbf{x}^N))^T$. Each marginal $p(\mathbf{f}_K^c)$ is Gaussian with mean and covariance given by $\mathbf{b}^c$ and $\mathbf{A}_c K_\nu(\mathbf{X},\mathbf{X})\mathbf{A}_c^T$, respectively. The covariance matrix between the pivot $\mathbf{f}_K^1$ and $\mathbf{f}_K^c$, for $c \neq 1$, is given by: $\mathbf{a}^1(\mathbf{a}^c)^T \odot K_\nu(\mathbf{X},\mathbf{X})$ with $\odot$ denoting Hadamart product. Proof given in App. A.

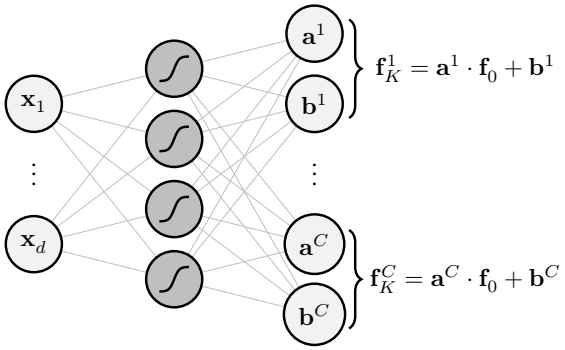

Figure 2. Architecture of the NN used. Hidden layers are shared, with the final layer giving each of the parameters of the C flows.

3.1. Approximate Inference

Our inference algorithm is inspired by the key observations of related works (Titsias, 2009; Hensman et al., 2013;

Maroñas et al., 2021). More precisely, we rely on a sparse VI algorithm where the variational distribution is defined so that the conditional's model prior $p(\mathbf{f}_K \mid \mathbf{u}_K)$ cancels (Titsias, 2009). We do not marginalize out the process values at the inducing points to allow for mini-batch optimization SVI (Hensman et al., 2013). Furthermore, we define the variational distribution over the GP space and then warp it with the same flows as the prior (Maroñas et al., 2021).

To start with, a set of M inducing points is defined on the GP space $f_0(\cdot)$. Note, however, that we can easily extend our framework to use inter-domain inducing points (Lázaro-Gredilla & Figueiras-Vidal, 2009), since we just need to derive the corresponding cross-covariances. Let $\overline{\mathbf{u}}_K$ be defined as $\overline{\mathbf{f}}_K$ in Eq. 6 summarizing the C transformed process values at the inducing points $\mathbf{Z}$. In App. A we show that the joint conditional prior is:

$$
\begin{aligned}
&p(\overline{\mathbf{f}}_K, \overline{\mathbf{u}}_K \mid \overline{\mathbf{W}}) = p(\mathbf{f}_0 \mid \mathbf{u}_0) \textstyle\prod_{k=0}^{K^1-1} |\det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}^1_k)}{\partial \mathbf{f}^1_k}|^{-1} \\
&\times \textstyle\prod_{c=2}^{C} \delta(\mathbf{f}^c_K - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c,\mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}^1_K)) \\
&\times p(\mathbf{u}_0) \textstyle\prod_{k=0}^{K^1-1} |\det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1,\mathbf{Z})}(\mathbf{u}^1_k)}{\partial \mathbf{u}^1_k}|^{-1} \\
&\times \textstyle\prod_{c=2}^{C} \delta(\mathbf{u}^c_K - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c,\mathbf{Z})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{Z})}(\mathbf{u}^1_K)). \qquad (7)
\end{aligned}
$$

This is possible because ETGP defines a valid stochastic process (i.e. consistent finite dimensional joint distribution), which means we can extend its finite dimensional distribution with inducing point locations as in standard GPs (Maroñas et al., 2021).

The variational distribution is assumed to have a similar form to the prior with some factors that are shared between them and others that are specific of the posterior approximation: $q(\overline{\mathbf{f}}_K, \overline{\mathbf{u}}_K, \overline{\mathbf{W}}) = q(\overline{\mathbf{f}}_K, \overline{\mathbf{u}}_K \mid \overline{\mathbf{W}}) q(\overline{\mathbf{W}})$ as in Maroñas et al. (2021), where the variational distribution over the NN has parameters $\overline{\phi}$ and is assumed to factorize across classes. Following Titsias (2009); Maroñas et al. (2021), the variational distribution over the values of the random processes at $\mathbf{X}$ and $\overline{\mathbf{Z}}$, $q(\overline{\mathbf{f}}_K, \overline{\mathbf{u}}_K) = p(\overline{\mathbf{f}}_K \mid \overline{\mathbf{u}}_K) q(\overline{\mathbf{u}}_K)$, is defined using the conditionals model's prior $p(\overline{\mathbf{f}}_K \mid \overline{\mathbf{u}}_K)$, given in Eq. 7, and a free form variational distribution $q(\overline{\mathbf{u}}_K)$. As in Maroñas et al. (2021), $q(\overline{\mathbf{u}}_K)$ is defined by warping a multivariate Gaussian defined on the original f_0 space $q(\mathbf{u}_0 \mid \mathbf{m}, \mathbf{S})$ using $\overline{\mathbb{G}}$, where $\mathbf{m} \in \mathbb{R}^M$, $\mathbf{S} \in \mathbb{R}^{M \times M}$ are the mean and covariance matrix variational parameters:

$$
\begin{aligned}
&q(\overline{\mathbf{u}}_K) = q(\mathbf{u}_0) \textstyle\prod_{k=0}^{K^1-1} |\det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1,\mathbf{Z})}(\mathbf{u}^1_k)}{\partial \mathbf{u}^1_k}|^{-1} \\
&\times \textstyle\prod_{c=2}^{C} \delta(\mathbf{u}^c_K - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c,\mathbf{Z})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{Z})}(\mathbf{u}^1_K)) \qquad (8)
\end{aligned}
$$

The resulting ELBO on the log-marginal-likelihood is, after several factor cancellations, equal to:

$$
\begin{aligned}
&\text{ELBO} = \Big\{ \textstyle\sum_{n=1}^{N} \sum_{c=1}^{C} \mathbb{I}(y^n = c) \mathbb{E}_{q(f_{0,n}) q(\overline{\mathbf{W}})} \\
&[\log \pi_c(\mathbb{G}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{x}^n)}(f_{0,n}), \ldots, \mathbb{G}^C_{\boldsymbol{\theta}(\mathbf{W}^C,\mathbf{x}^n)}(f_{0,n}))] \Big\} \\
&- \text{KL}[q(\mathbf{u}_0) || p(\mathbf{u}_0)] - \textstyle\sum_{c=1}^{C} \text{KL}[q(\mathbf{W}^c) || p(\mathbf{W}^c))], \qquad (9)
\end{aligned}
$$

where $q(f_0)$ is computed as in Sec. 2.1. The expectation w.r.t. $q(\overline{\mathbf{W}})$ is computed via Monte Carlo and a single 1-d quadrature is used to compute expectations over $q(f_0)$. $\text{KL}[q(\mathbf{u}_0) \,||\, p(\mathbf{u}_0)]$ is tractable. $\text{KL}[q(\mathbf{W}^c) \,||\, p(\mathbf{W}^c))]$ can be computed in closed form for certain choices of the prior and variational posterior. However, we follow Maroñas et al. (2021) and use Monte Carlo Dropout (Gal & Ghahramani, 2016) (MCDROP) rather than VI to perform inference on $\mathbf{W}$. With this, the same NN can be used to make non-Bayesian point estimate predictions (PE-ETGP) (Srivastava et al., 2014) or Bayesian predictions (BA-ETGP) (Gal & Ghahramani, 2016). This objective can be maximized using stochastic optimization methods and the data can be sub-sampled for mini-batch training. Readers concerned with the cancellation of delta functions in Eq. 9 can replace them with Gaussians with variance σ^2 to then take the limit $\sigma^2 \to 0$.

Predictions for $y^\star$ associated to a new $\mathbf{x}^\star$ are computed using an approximate predictive distribution:

$$
\begin{aligned}
&p(y^\star \mid \mathbf{x}^\star, \mathcal{D}) \approx \mathbb{E}_{q(f_0(\mathbf{x}^\star)) q(\overline{\mathbf{W}})} [p(y^\star \mid \mathbb{G}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{x}^\star)}(f_0(\mathbf{x}^\star)), \\
&\ldots, \mathbb{G}^C_{\boldsymbol{\theta}(\mathbf{W}^C,\mathbf{x}^\star)}(f_0(\mathbf{x}^\star)))], \qquad (10)
\end{aligned}
$$

where the integral is approximated by Monte Carlo and 1-d quadrature methods, after having marginalized out $\overline{\mathbf{u}}_K$.

3.2. Summary of the Proposed Method and Computational Cost

ETGP creates C dependent processes since $\mathbf{f}_0$ is shared. We have characterized the dependencies of these C processes for a linear flow in Prop. 3.1. The flows, however, need not be linear and can be arbitrarily complicated. Because $\mathbb{G}^c_{\theta_K}$ is input-dependent the C processes are also non-stationary.

Expectations w.r.t. the NN's parameters can be computed using batched matrix multiplications. Expectations w.r.t. $q(\mathbf{f}_0)$ in Eq. 9 and Eq. 10 can be computed with 1-d quadrature. By contrast, the SVGP method from Sec. 2.1 cannot use quadrature methods since it considers C independent processes. Moreover, the number of GP operations in ETGP is constant with C. To get $q(\mathbf{f}^c)$ in SVGPs Sec. 2.1 one needs a cubic operation to invert $K^c_\nu(\mathbf{z}, \mathbf{z})$ and M^2 operation to compute the variational parameters per class and datapoint, giving a complexity of $\mathcal{O}(CM^3 + CNM^2)$. This can be alleviated by sharing K_ν and $\mathbf{Z}$ across GPs, resulting in $\mathcal{O}(M^3 + CNM^2)$, at the cost of limiting expressiveness, as we'll show. ETGP cost is always $\mathcal{O}(M^3 + NM^2)$

(without considering the NN's computations, which for the architecture presented is often much faster and can be done in parallel to GP operations).

4. Related Work

To model non-stationary data, the typical approach is to use non-stationary covariance functions such as the Neural Network (Williams, 1996) or the Arcosine (Cho & Saul, 2009). Our experiments, however, show that ETGP provides superior results in the multi-class setting when compared to a method using these kernels. One can also make stationary kernels non-stationary by making the parameters of the kernel depend on the input (Heinonen et al., 2016). However, the work in (Heinonen et al., 2016) is limited to small datasets since it does not consider sparse GPs and it relies on Hamilton Monte Carlo for approximate inference, which is computationally expensive. A sparse GP approach would require a GP per kernel hyperparameter which can lead to a big number of GPs for high d. Another approach to obtain non-stationary processes considers processes mixing using hierarchical models (Wang et al., 2020), One can also place GPs over the mixing matrix entries, achieving input-dependent length scales and amplitudes (Wilson et al., 2012). These works either don't scale for high C (Wilson et al., 2012) or require domain knowledge to avoid misspecification (Wang et al., 2020).

Non-stationary data can also be modeled by warping the input space. More precisely, one can use a non-linear transformation before introducing the data into the kernel (Sampson & Guttorp, 1992; Schmidt et al., 2000; Calandra et al., 2016; Wilson et al., 2016). These methods, however, either run the risk of over-fitting, as a consequence of not regularizing the parameters of the non-linear transformation nor using a fully Bayesian approach, or either do not scale to large datasets.

One can also use more sophisticated processes such as the GP Product Model (Adams & Stegle, 2008), DGPs (Damianou & Lawrence, 2013) or TGPs (Maroñas et al., 2021) to achieve non-stationarity. The GP Product Model, however, does not scale to large datasets. DGPs have been shown to give similar results to those of TGPs. However, TGPs have a lower computational cost than DGPs and slightly higher than SVGPs. Therefore, the proposed method, ETGP, is expected to be faster than TGPs and also faster than DGPs, in consequence.

From the dependence point of view, several approaches have been considered, which range from using process convolutions (Boyle & Frean, 2004), to combining latent GPs via a mixing matrix (Álvarez et al., 2012) whose entries can be parameterized by a GP (Wilson et al., 2012). More recently, (Jankowiak & Gardner, 2019) extends Multi-output GPs (Álvarez et al., 2012), Gaussian Process Regression Networks (Wilson et al., 2012) and DGP (Damianou & Lawrence, 2013) by using NNs to replace different building blocks of these methods. Since the computational cost of considering several GPs for inference is high (not necessarily for multi-class learning), several methods have tried to alleviate this cost by, *e.g.*, using sparse methods (Alvarez & Lawrence, 2008) or more recently by assuming that the data lives around a linear subspace and then exploit a low-rank structure of the covariance matrix (Bruinsma et al., 2020). All these works, however, use several GPs for modeling the data, unlike the proposed method ETGP, and are hence expected to be significantly more expensive computationally.

5. Experiments

We evaluate ETGP in 5 UCI datasets (Lichman, 2013) (see Fig. 3 for details). We compare ETGP with LINEAR, SAL and TANH flows with Bayesian (BA-ETGP) and point estimate (PE-ETGP) predictions. We compare against a stationary independent (RBF), as described in Sec. 2.1, and dependent (RBFCORR) SVGPs, where dependencies are obtained by mixing C latent GPs (Álvarez et al., 2012). We also compare against two non-stationary SVGP with an arccosine (ARCCOS) (Cho & Saul, 2009) and a Neural Network (NNET) (Williams, 1996) kernels. In SVGP, we run the model with separate/shared kernels and inducing points for each class label, indicated with separate/shared K_ν in the results. We report accuracy (ACC) here. App. B contains test log-likelihood (LL) results and gives all training details. We highlight that on each SVGP run (one per training hyperparameters), we pick the best result on the *test set*, so that the comparison with the ETGP is the most optimistic. By contrast, we perform model selection with a validation set for ETGP. The code will be released in this repository `https://github.com/jmaronas/Efficient_Multiclass_Gaussian_Processes_using_TGP`[3].

5.1. Prediction Performance Analysis

We compare ETGP in terms of prediction performance against stationary dependent/independent and non-stationary SVGPs. The results obtained are displayed in Fig. 3 (and Fig. 9 in App. B.) We observe that on the big datasets with a large number of classes C (*i.e.*, `characterfont` and `devangari`) ETGP clearly outperforms or gives comparable results to SVGPs, but around one order of magnitude faster, see Sec. 5.2 and Fig. 4. In the worst case, the ACC gain goes from 0.34 RBFCORR to 0.36 TANH. In the best case, we see a boost from 0.29 SVGP to 0.36 TANH in `characterfont`. In `devangari` ETGPs are clearly better in all cases boosting accuracy from 0.93 RBFCORR to

[3]This repository is being used for other projects so at the moment stays closed. Drop us an email and we will happily share the code.

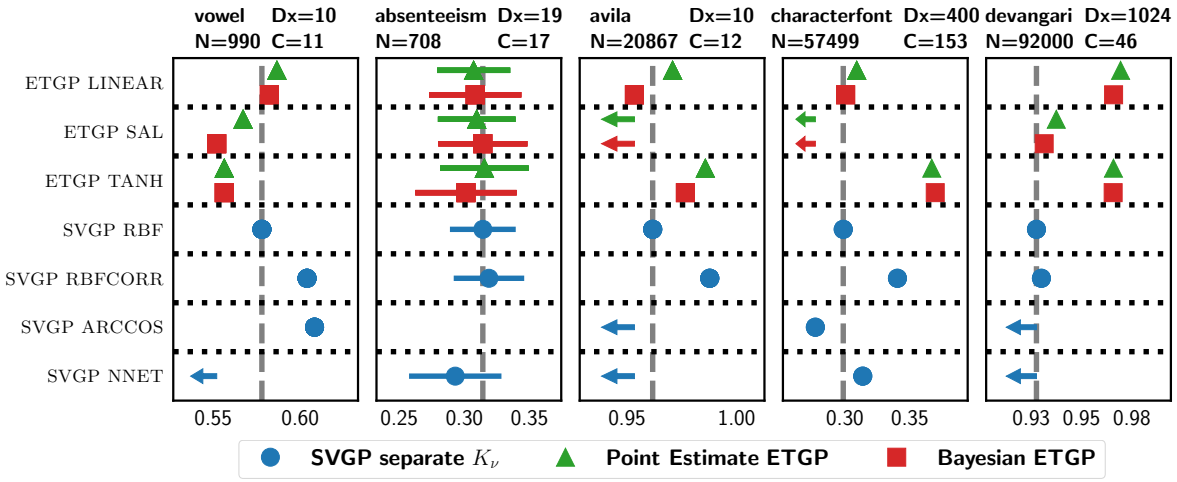

Figure 3. Accuracy (right is better) comparing ETGP vs. independent/dependent stationary GPs. and non-stationary

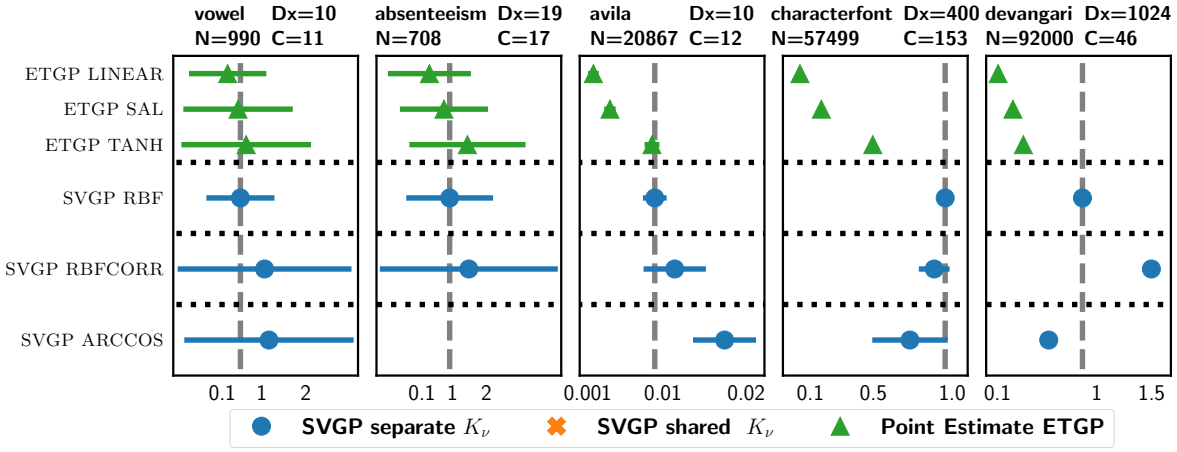

Figure 4. Average training time per epoch in minutes (left is better) comparing ETGP with SVGPs. NNET kernel is omitted as it is slower. Times for `absenteeism` and `vowel` are scaled by 10^3.

0.96 TANH and LINEAR, nearly matching the result obtained by a convolutional neural network (0.98) (Acharya et al., 2015). This increment in performance is obtained around one order of magnitude faster as well. In `avila`, a medium size dataset, ETGP (0.985 TANH 0.970 LINEAR) works better than SVGP (0.962) and comparable to correlated SVGP (0.988). However, again, ETGP is one order of magnitude faster. In `vowel` we observe SVGP working better than ETGP, being the model with shared kernel the best (see App. B). Further analysis is provided in the appendix but we observe it's due to domain shift between training and test sets and due to the way models are selected for the SVGP. On `absenteeism` we observe that ETGP works similarly to SVGP. However, ETGP is much faster to train. Finally, across all datasets, the SAL flow is the worst one (see App. B for possible explanations). This also suggests that the results from Maroñas et al. (2021) can be improved since they only used SAL flow in their non-stationary TGPs. We remark the good performance of ETGP LINEAR, which defines C non-stationary dependent GPs, opening its use in other applications.

Regarding LL (see Fig. 9 in App. B), we observe similar results across all datasets. For small datasets BA-ETGP provides much better uncertainty quantification (LL) than its non-Bayesian counterpart ETGP. This is particularly the case in `vowel`, supporting our previous justification about the decrease in performance of ETGP relative to SVGP in this dataset. In our experiments we observed that the models trained with higher dropout probability worked even better in the small datasets, since they are incorporating more epistemic uncertainty. However, this boost is not observed in the medium/large datasets in either ACC or LL. This matches findings from Maroñas et al. (2021) where being Bayesian

does not show improvements in classification datasets. This is expected as with big N epistemic uncertainty vanishes, however for `vowel` and due to its domain-shift between training and test set, a model with higher dropout probability is not being selected. Overall, this suggests that alternatives that fix the dropout probability depending on the number of training points, such as concrete dropout (Gal et al., 2017), are a potential line of research to enhance the performance of ETGP.

Regarding comparison against non-stationary kernels, we do not observe a general tendency in the way they perform. This remarks the importance of having background knowledge about the non-stationarity of the particular task. Our work and Maroñas et al. (2021) show that the non-stationarity achieved by input-dependent flows is beneficial and easy to interpret since we just make each of the marginals depend directly on the part of the feature space that we are modeling, with no cross interactions between data points beyond those given by the base stationary kernel. We do not report results for ARCCOS on `absenteeism` as we found training runs to saturate numerically (using float64 precision).

5.2. Training Timing Comparison

We report average training time in Fig. 4 and prediction time in App. B. By using MCDROP, the training time of the PE-ETGP and BA-ETGP is the same. Predictions for the Bayesian ETGPs can be computed in parallel. We observe that ETGP is the fastest method, with a gain, in many cases, of one order of magnitude, compared to SVGP. SVGP with shared K_ν is competitive with ETGP in terms of training time, but has a drop in performance (see Sec. 5.3), unlike ETGP which typically performs better.

5.3. Comparison with SVGPs with Shared Kernel and Inducing Points and with DGPs

We observe that sharing K_ν and $\mathbf{Z}$ across the C GPs clearly drops performance as shown in Fig. 5 for accuracy and also for log-likelihood (see App. B). We've refactorized GPFLOW's source code so that the shared correlated SVGP

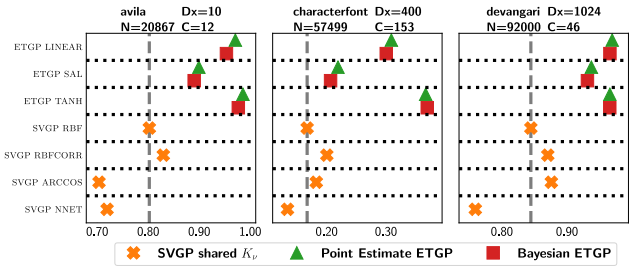

Figure 5. Accuracy (right is better) comparing ETGP vs. SVGPs with shared kernels.

Table 1. Results for 2 layers DGP.

	Accuracy	Log Likelihood	Training Time
`avila`	0.888	−0.310	0.008
`characterfont`	0.171	−3.636	0.197
`devangari`	0.909	−0.369	0.177

Table 2. Ablation study comparing TGP and ETGP trained (not) freezing the GP parameters.

		`avila`		`characterfont`		`devangari`	
		ACC	LL	ACC	LL	ACC	LL
LINEAR	ETGP Frozen	0.951	−0.146	0.269	−3.360	0.968	−0.133
	ETGP	0.973	−0.089	0.301	−3.268	0.969	−0.132
	TGP Frozen	0.956	−0.139	0.269	−3.374	0.969	−0.131
	TGP	0.990	−0.044	0.327	−3.185	0.969	−0.132
TANH	ETGP Frozen	0.986	−0.050	0.373	−3.802	0.967	−0.173
	ETGP	0.983	−0.055	0.378	−3.789	0.968	−0.168
	TGP Frozen	0.985	−0.052	0.373	−3.688	0.966	−0.179
	TGP	0.991	−0.029	0.376	−3.756	0.967	−0.179

model is more efficient, see App. C. This drop in performance affects not only SVGPs but also a 2 layer's DGPs with the configuration from Salimbeni & Deisenroth (2017), which shares K_ν and $\mathbf{Z}$ across layers, as shown in Tab. 1. Also, some of these models needed considerably more RAM memory than the SVGP or ETGP. The results for `vowel` and `absenteeism` are discussed in App. B. We also observe that the training time of these shallow DGPs is superior to ETGP.

5.4. Ablation Study and Comparison with TGP

We perform an ablation study that shows that the expressiveness of the ETGP and TGP not only comes from the BNN but also from the GP. In other words, the ETGP cannot be reduced to a model which could simply transform a white noise process. We train several ETGPs and TGPs freezing the parameters from the GP during optimization. Results in Tab. 2 shows that the GP part is relevant and that TGP is usually more expressive than ETGP, as expected. Unsurprisingly there is less difference between freezing and not freezing the ETGP for the more expressive flow (TANH). This suggests that in a real application, we could find a trade-off between having a more expressive flow at the benefit of not training the GP at all, or having a less expressive one, since more expressive flows usually require higher computations[4]. This would be a new stochastic process specified by transforming a white-noise process, which we let for future work.

5.5. Comparison with Point Estimate Neural Networks

As suggested by one of the reviewers we run experiments comparing with Neural Networks. In particular, we run two-layers fully connected neural network with Relu activation

[4]Computations in this work can be speed up by running the BNN in a GPU using a different computational thread so the computations involved (GP and BNN) are triggered in parallel.

Table 3. Results comparing against DNNs.

	Accuracy	Log Likelihood
characterfont	0.418	−2.809
devangari	0.974	−0.125

and a dropout probability of 0.5. We have optimized these networks using different training hyperparameters and we report those that obtained the best result on the test set following our experimental procedure on the baselines.

We have avoided comparison with vowel and absenteeism since these datasets have very few training points and hence a DNN typically over-fits (Hernández-Lobato & Adams, 2015). We also avoid avila since it is a tabular dataset where it is known that DNNs perform poorly and require specialized architectures (Du et al., 2021).

Results for characterfont and devangari are provided in Tab. 3. As mentioned before a Convolutional Neural Network obtains 0.98 ACC on devangari. We observe that fully connected networks perform either better (characterfont) or comparable (devangari) to the method presented. This is to be expected as these are image datasets with a similar structure to MNIST, where a Fully Connected Neural Network works often better than a standard GP, as reported in the literature (see, e.g., (van der Wilk et al., 2017)). Notwithstanding, again, we would like to remark that the results above for characterfont and devangari are too optimistic and an upper bound of the actual results of the DNN, since selection is done by looking at test metrics.

6. Conclusions and Future Work

We have introduced the Efficient Transformed Gaussian Process (ETGP) for creating C dependent non-stationary stochastic processes in an efficient way. For this, a single initial GP is transformed C times using (non-)linear invertible transformations. This has the benefit of reducing the computational cost while providing enough model flexibility to learn complex tasks. We have provided an efficient training algorithm for ETGP based on variational inference. This method has been evaluated in the context of multi-class classification. Our results show that ETGP is competitive or even better than typical sparse SVGPs, at a lower computational cost. A limitation of ETGP is, however, that it leads to a model that is more difficult to interpret than SVGPs, as a consequence of the non-linear transformations. However, particular flows may allow controlling the moments of the induced distributions, improving interpretability (Rios & Tobar, 2019).

Future work may focus on extending ETGP to multi-task and multi-label problems, where a large number of processes may also be needed. The use of ETGP in a deep kernel learning framework can also be investigated (Wilson et al., 2016). Moreover, an analysis of how the dependencies among process values are modeled by sharing the copula of the base GP can be carried out. Additional dependencies can be incorporated by further mixing each latent function in ETGP using a mixing matrix. The use of concrete dropout and GPs instead of NNs to parameterize each flow are also alternatives to improve different aspects of ETGP, such as computational performance, well-specified Bayesian priors, and epistemic uncertainty. Their use is left for future work.

Societal Impact

This work introduces a new probabilistic machine learning method that can be used to make predictions quantifying its uncertainty. Uncertainty quantification makes probabilistic methods particularly suited to be deployed in high-risk scenarios such as autonomous driving or medicine. In all these problems it is required the model to be interpretable and to provide reliable predictions. As a consequence, a careful analysis of the correct modeling of these properties must be done before deploying the model to avoid potential negative societal impacts.

Also, one needs to consider the particular limitations of the approximate inference algorithm and model assumptions on each particular application. For example, the presented inference algorithm can only be used with diagonal flows, which leaves the dependencies (copula) of the GP unchanged. Thus, if one believes the copula is non-Gaussian, deploying ETGP can result in biased predictions that may have harmful consequences.

Acknowledgements

The authors acknowledge funding coming from the Spanish Plan Nacional I+D+I, project PID2019-106827GB-I00 / AEI / 10.13039/501100011033. We also acknowledge the use of the computational resources from Centro de Computación Científica (CCC) and the AUDIAS Laboratory both at Universidad Autónoma de Madrid. We would like to thank anonymous reviewers since their comments contributed to improving the paper's legibility. Part of this work was carried out while J.M. was at the PRHLT Research Center at Universidad Politécnica de Valencia.

References

Acharya, S., Pant, A. K., and Gyawali, P. K. Deep learning based large scale handwritten Devanagari character recognition. *2015 9th International Conference on Software, Knowledge, Information Management and Applications*

(SKIMA), pp. 1–6, 2015.

Adams, R. P. and Stegle, O. Gaussian process product models for nonparametric nonstationarity. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 1–8, 2008.

Alvarez, M. and Lawrence, N. Sparse convolved Gaussian processes for multi-output regression. In *Advances in Neural Information Processing Systems*, pp. 57–64, 2008.

Álvarez, M. A., Rosasco, L., and Lawrence, N. D. Kernels for vector-valued functions: A review. *Found. Trends Mach. Learn.*, 4:195–266, 2012.

Bishop, C. M. *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer, 2007.

Blackford, L. S., Petitet, A., Pozo, R., Remington, K., Whaley, R. C., Demmel, J., Dongarra, J., Duff, I., Hammarling, S., Henry, G., et al. An updated set of basic linear algebra subprograms (BLAS). *ACM Transactions on Mathematical Software*, 28:135–151, 2002.

Boyle, P. and Frean, M. Dependent Gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 217–224, 2004.

Bruinsma, W., Perim, E., Tebbutt, W., Hosking, S., Solin, A., and Turner, R. Scalable exact inference in multi-output Gaussian processes. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 1190–1201, 2020.

Calandra, R., Peters, J., Rasmussen, C. E., and Deisenroth, M. P. Manifold Gaussian processes for regression. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 3338–3345, 2016.

Chai, K. M. A. Variational multinomial logit Gaussian process. *The Journal of Machine Learning Research*, 13: 1745–1808, 2012.

Cho, Y. and Saul, L. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems*, pp. 342–350, 2009.

Damianou, A. and Lawrence, N. Deep Gaussian processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, pp. 207–215, 2013.

Du, L., Gao, F., Chen, X., Jia, R., Wang, J., Zhang, J., Han, S., and Zhang, D. Tabularnet: A neural network architecture for understanding semantic structures of tabular data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '21, pp. 322–331, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383325.

Duvenaud, D., Lloyd, J., Grosse, R., Tenenbaum, J., and Zoubin, G. Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning*, pp. 1166–1174, 2013.

Gal, Y. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

Gal, Y. and Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, pp. 1050–1059, 2016.

Gal, Y., Hron, J., and Kendall, A. Concrete dropout. In *Advances in Neural Information Processing Systems*, volume 30, pp. 3581–3590, 2017.

Gardner, J. R., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. Gpytorch: Blackbox matrix-matrix gaussian process inference with GPU acceleration. In *NeurIPS*, pp. 7587–7597, 2018.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 06–11 Aug 2017.

Hamelijnck, O., Damoulas, T., Wang, K., and Girolami, M. Multi-resolution multi-task Gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 14025–14035, 2019.

Heinonen, M., Mannerström, H., Rousu, J., Kaski, S., and Lähdesmäki, H. Non-stationary Gaussian process regression with Hamiltonian monte carlo. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pp. 732–740, 2016.

Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pp. 282–290, 2013.

Hensman, J., de G. Matthews, A. G., and Ghahramani, Z. Scalable variational gaussian process classification. In *AISTATS*, JMLR Workshop and Conference Proceedings, 2015.

Hernández-Lobato, J. M. and Adams, R. P. Probabilistic backpropagation for scalable learning of bayesian neural

networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 1861–1869. JMLR.org, 2015.

Jankowiak, M. and Gardner, J. Neural likelihoods for multi-output Gaussian processes, 2019.

Krige, D. G. *A statistical approach to some mine valuation and allied problems on the Witwatersrand: by DG Krige*. PhD thesis, University of the Witwatersrand, 1951.

Lawrence, N. Gaussian process latent variable models for visualisation of high dimensional data. In *Advances in neural information processing systems*, pp. 329–336, 2003.

Lázaro-Gredilla, M. and Figueiras-Vidal, A. Inter-domain Gaussian processes for sparse inference using inducing features. In *Advances in Neural Information Processing Systems 22*, pp. 1087–1095, 2009.

Leimkuhler, B. and Matthews, C. *Molecular Dynamics: With Deterministic and Stochastic Numerical Methods*. Interdisciplinary Applied Mathematics. Springer International Publishing, 2015.

Lichman, M. UCI machine learning repository, 2013.

Maroñas, J., Hamelijnck, O., Knoblauch, J., and Damoulas, T. Transforming Gaussian processes with normalizing flows. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, pp. 1081–1089, 2021.

Matthews, A. G. d. G., van der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., León-Villagrá, P., Ghahramani, Z., and Hensman, J. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18:1–6, 2017.

Neal, R. M. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1996.

Petersen, K. B. and Pedersen, M. S. The matrix cookbook, 2012. Version 20121115.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006.

Rios, G. Transport Gaussian processes for regression, 2020.

Rios, G. and Tobar, F. Compositionally-warped Gaussian processes. *Neural Networks*, 118:235–246, 2019.

Salimbeni, H. and Deisenroth, M. Doubly stochastic variational inference for deep gaussian processes. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Sampson, P. D. and Guttorp, P. Nonparametric estimation of nonstationary spatial covariance structure. *Journal of the American Statistical Association*, 87:pp. 108–119, 1992.

Schmidt, A. M., O'Hagan, A., and Schmidt, R. M. Bayesian inference for nonstationary spatial covariance structure via spatial deformations. *Journal of the Royal Statistical Society, Series B*, 65:745–758, 2000.

Sklar, A. Fonctions de répartition à n dimensions et leurs marges. *Publications de l'Institut de Statistique de l'Université de Paris*, 8:229–231, 1959.

Snelson, E. L., Rasmussen, C. E., and Ghahramani, Z. Warped gaussian processes. In *NIPS*, pp. 337–344, 2003.

Snoek, J., Larochelle, H., and Adams, R. Practical Bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pp. 2951–2959, 2012.

Snoek, J., Swersky, K., Zemel, R., and Adams, R. P. Input warping for Bayesian optimization of non-stationary functions. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, pp. II–1674–II–1682, 2014.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

Titsias, M. Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, pp. 567–574, 2009.

Uhlenbeck, G. E. and Ornstein, L. S. On the theory of the Brownian motion. *Physical review*, 36:823, 1930.

van der Wilk, M., Rasmussen, C. E., and Hensman, J. Convolutional gaussian processes. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

van der Wilk, M., Dutordoir, V., John, S., Artemev, A., Adam, V., and Hensman, J. A framework for interdomain and multioutput Gaussian processes, 2020.

Wang, K., Hamelijnck, O., Damoulas, T., and Steel, M. Non-separable non-stationary random fields. In *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pp. 9887–9897, 2020.

Williams, C. Computing with infinite networks. In *Advances in Neural Information Processing Systems*, pp. 295–301, 1996.

Wilson, A. G. and Ghahramani, Z. Copula processes. In Lafferty, J. D., Williams, C. K. I., Shawe-Taylor, J., Zemel, R. S., and Culotta, A. (eds.), *Advances in Neural Information Processing Systems 23*, pp. 2460–2468, 2010.

Wilson, A. G., Knowles, D. A., and Ghahramani, Z. Gaussian process regression networks. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pp. 1139–1146, 2012.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pp. 370–378, 2016.

Yang, G. Wide feedforward or recurrent neural networks of any architecture are Gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 9947–9960, 2019.

A. Math Appendix

In this appendix we provide a wider description of the equations involved in this paper, ranging from the model definition to the sparse variational inference algorithm. For completeness, and to make the improvements of the proposed model clearer, we start by describing the sparse variational inference algorithm applied to multi-class problems with and independent GP prior, previous to the definition of our model. Although we compare against correlated GPS using a mixing matrix, we don't provide its derivation since it goes beyond the scope of this appendix. In the end of this appendix we proof Prop. 3.1. If possible we keep all the conditioning set explicit in the equations presented.

A.1. Classification with Independent GP Priors

Given a classification problem with inputs $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ and C outputs $y \in \mathcal{Y} \subset \mathbb{N}$, we want to learn C mapping functions from $\mathbf{x}$ to the probability of belonging to each y; given a set of observations $\mathcal{D} = \{\mathbf{x}^n, y^n\}_{n=1}^N$ with $\mathbf{X} = (\mathbf{x}^1, \dots, \mathbf{x}^N)$ and $\mathbf{y} = (y^1, \dots, y^N)$.

If this modeling procedure is done using GPS, then we place an independent GP on each of these C functions, each one parameterized by a mean function $\mu_\nu(\mathbf{x})$ (which we assume to be zero) and a covariance matrix $K_\nu^c(\mathbf{x}, \mathbf{x})$, parameterized by ν. Following the notation introduced in the main paper, the joint distribution of C processes at locations $\mathbf{X}$ is given by:

$$p(\overline{\mathbf{f}}_0 \mid \overline{\nu}) = \prod_{c=1}^{C} \mathcal{N}\left(\mathbf{f}_0^c \mid \mathbf{0}, K_\nu^c(\mathbf{X}, \mathbf{X})\right) \tag{11}$$

where $\overline{\mathbf{f}}_0 = \{\mathbf{f}_0^1, \dots, \mathbf{f}_0^C\}$. This prior is combined with a likelihood $p(\mathbf{y} \mid \overline{\mathbf{f}}_0)$ that links latent functions to observations. In classification, a common choice is the Categorical Likelihood, giving the joint distribution:

$$p(\mathbf{y}, \overline{\mathbf{f}}_0) = p(\mathbf{y} \mid \overline{\mathbf{f}}_0) p(\overline{\mathbf{f}}_0) = \left[\prod_{n=1}^{N} \prod_{c=1}^{C} \pi_c(\overline{f}_{0,n})^{\mathbb{I}(y^n = c)}\right] \prod_{c=1}^{C} \mathcal{N}\left(\mathbf{f}_0^c \mid \mathbf{0}, K_\nu^c(\mathbf{X}, \mathbf{X})\right) , \tag{12}$$

where $\pi_c(\overline{f}_{0,n}) = \exp(f_0^c(\mathbf{x}^n)) / \sum_{c'=1}^{C} \exp(f_0^{c'}(\mathbf{x}^n))$ is the Softmax link function mapping latent vectors to probabilities, and $\mathbb{I}(\cdot)$ the indicator function.

In Bayesian learning, we are interested in the posterior $p(\mathbf{f}_0 \mid \mathcal{D})$, which is intractable for many likelihoods, and computationally unfeasible since its complexity scales cubically with the number of training points. We now introduce the variational sparse derivation.

A.1.1. Variational Sparse Derivation

The idea behind sparse GPS is to use a set of M inducing points $\mathbf{z} \in \mathcal{X}$, with $\mathbf{Z} = (\mathbf{z}^1, \dots, \mathbf{z}^M)$, that acts as summary statistics of the data. Each inducing point $\mathbf{z}$ has an associated latent value u_0. Following the GP's prior definition, then at $\mathbf{Z}$ we have $\mathbf{u}_0 \sim \text{GP}(\mathbf{u}_0 \mid 0, K_\nu(\mathbf{Z}, \mathbf{Z})))$. Thus, given $\mathbf{X}, \mathbf{Z}$ the joint distribution is Gaussian given by:

$$p(\mathbf{f}_0, \mathbf{u}_0 \mid \mathbf{X}, \mathbf{Z}, \nu) = \mathcal{N}\left(\begin{array}{c|cc} \mathbf{f}_0 & \mathbf{0} & K_\nu(\mathbf{X}, \mathbf{X}), \; K_\nu(\mathbf{X}, \mathbf{Z}) \\ \mathbf{u}_0 & \mathbf{0} & K_\nu(\mathbf{Z}, \mathbf{X}), \; K_\nu(\mathbf{Z}, \mathbf{Z}) \end{array}\right) \tag{13}$$

One key contribution from Titsias (2009) is to define these inducing points to be variational parameters that are learned by minimizing the KL between the approximate $q(\mathbf{f}_0, \mathbf{u}_0)$ and true posterior $p(\mathbf{f}_0, \mathbf{u}_0 \mid \mathcal{D}, \mathbf{Z}, \nu)$. Since $\mathbf{z}$ do not belong to the model parameters, then they don't increase the model expressiveness hence protecting the learning procedure from overfitting.

The other key contribution from Titsias (2009) is how the variational distribution is defined. In particular, $q(\mathbf{f}_0, \mathbf{u}_0) = p(\mathbf{f}_0 \mid \mathbf{u}_0) q(\mathbf{u}_0)$ so that the conditional's model prior $p(\mathbf{f}_0 \mid \mathbf{u}_0)$ gets canceled. Later, Hensman et al. (2013) propose to keep $q(\mathbf{u}_0 \mid \mathbf{m}, \mathbf{S})$ explicit by parameterizing it with a Gaussian distribution with parameters $\mathbf{m} \in \mathbb{R}^M$, $\mathbf{S} \in \mathbb{R}^{M \times M}$. With this, the ELBO can be optimized with stochastic variational inference.

If C independent GPS are going to be used, we can easily extend the presented equations as follows. The joint prior factorizes

across C as above:

$$p(\overline{\mathbf{f}}_0, \overline{\mathbf{u}}_0 \mid \mathbf{X}, \overline{\mathbf{Z}}, \overline{\nu}) = \prod_{c=1}^{C} \mathcal{N}\left(\begin{array}{c|cc} \mathbf{f}_0^c & \mathbf{0} & K_\nu^c(\mathbf{X},\mathbf{X}),\ K_\nu(\mathbf{X},\mathbf{Z}^c) \\ \mathbf{u}_0^c & \mathbf{0} & K_\nu^c(\mathbf{Z}^c,\mathbf{X}),\ K_\nu(\mathbf{Z}^c,\mathbf{Z}^c) \end{array} \right) \tag{14}$$

and the approximate posterior can be defined by factorizing across C as well:

$$q(\overline{\mathbf{f}}_0, \overline{\mathbf{u}}_0 \mid \mathbf{X}, \overline{\mathbf{Z}}, \overline{\nu}, \overline{\mathbf{m}}, \overline{\mathbf{S}}) = \prod_{c=1}^{C} p(\mathbf{f}_0^c \mid \mathbf{u}_0^c, \mathbf{X}, \mathbf{Z}^c, \nu_c) q(\mathbf{u}_0^c \mid \mathbf{m}^c, \mathbf{S}^c) \tag{15}$$

The KL minimization between the approximate posterior and the true posterior is equivalent to maximizing the Evidence Lower Bound (ELBO), which we now derive:

$$\begin{aligned} \text{ELBO} &= \int_{\mathbf{f}_0^1} \dots \int_{\mathbf{f}_0^C} \int_{\mathbf{u}_0^1} \dots \int_{\mathbf{u}_0^C} q(\overline{\mathbf{f}}_0, \overline{\mathbf{u}}_0) \log \frac{p(\mathbf{y} \mid \overline{\mathbf{f}}_0) p(\overline{\mathbf{f}}_0, \overline{\mathbf{u}}_0)}{q(\overline{\mathbf{f}}_0, \overline{\mathbf{u}}_0)} \mathrm{d}\mathbf{f}_0^1 \dots \mathrm{d}\mathbf{f}_0^C \mathrm{d}\mathbf{u}_0^1 \dots \mathrm{d}\mathbf{u}_0^C \\ &= \underbrace{\int_{\mathbf{f}_0^1} \dots \int_{\mathbf{f}_0^C} \int_{\mathbf{u}_0^1} \dots \int_{\mathbf{u}_0^C} q(\overline{\mathbf{f}}_0, \overline{\mathbf{u}}_0) \log p(\mathbf{y} \mid \overline{\mathbf{f}}_0) \mathrm{d}\mathbf{f}_0^1 \dots \mathrm{d}\mathbf{f}_0^C \mathrm{d}\mathbf{u}_0^1 \dots \mathrm{d}\mathbf{u}_0^C}_{\text{ELL}} \\ &\underbrace{+ \int_{\mathbf{f}_0^1} \dots \int_{\mathbf{f}_0^C} \int_{\mathbf{u}_0^1} \dots \int_{\mathbf{u}_0^C} q(\overline{\mathbf{f}}_0, \overline{\mathbf{u}}_0) \log \frac{p(\overline{\mathbf{f}}_0, \overline{\mathbf{u}}_0)}{q(\overline{\mathbf{f}}_0, \overline{\mathbf{u}}_0)} \mathrm{d}\mathbf{f}_0^1 \dots \mathrm{d}\mathbf{f}_0^C \mathrm{d}\mathbf{u}_0^1 \dots \mathrm{d}\mathbf{u}_0^C}_{-\text{KL}} \end{aligned} \tag{16}$$

where we have dropped the conditioning set for clarity. We now workout each term separately:

$$\begin{aligned} -\text{KL} &= \int_{\mathbf{f}_0^1} \dots \int_{\mathbf{f}_0^C} \int_{\mathbf{u}_0^1} \dots \int_{\mathbf{u}_0^C} q(\overline{\mathbf{f}}_0, \overline{\mathbf{u}}_0) \log \frac{p(\overline{\mathbf{f}}_0, \overline{\mathbf{u}}_0)}{q(\overline{\mathbf{f}}_0, \overline{\mathbf{u}}_0)} \mathrm{d}\mathbf{f}_0^1 \dots \mathrm{d}\mathbf{f}_0^C \mathrm{d}\mathbf{u}_0^1 \dots \mathrm{d}\mathbf{u}_0^C \\ &= \int_{\mathbf{u}_0^1} \dots \int_{\mathbf{u}_0^C} q(\overline{\mathbf{u}}_0) \log \frac{\cancel{p(\overline{\mathbf{f}}_0 \mid \overline{\mathbf{u}}_0)} p(\overline{\mathbf{u}}_0)}{\cancel{p(\overline{\mathbf{f}}_0 \mid \overline{\mathbf{u}}_0)} q(\overline{\mathbf{u}}_0)} \mathrm{d}\mathbf{u}_0^1 \dots \mathrm{d}\mathbf{u}_0^C \\ &= \int_{\mathbf{u}_0^1} \dots \int_{\mathbf{u}_0^C} \prod_{c=1}^{C} q(\mathbf{u}_0^c) \sum_{c'=1}^{C} \log \frac{p(\mathbf{u}_0^{c'})}{q(\mathbf{u}_0^{c'})} \mathrm{d}\mathbf{u}_0^1 \dots \mathrm{d}\mathbf{u}_0^C = \\ &= \sum_{c'=1}^{C} \int_{\mathbf{u}_0^1} \dots \int_{\mathbf{u}_0^C} \prod_{c=1}^{C} q(\mathbf{u}_0^c) \log \frac{p(\mathbf{u}_0^{c'})}{q(\mathbf{u}_0^{c'})} \mathrm{d}\mathbf{u}_0^1 \dots \mathrm{d}\mathbf{u}_0^C = \\ &= \sum_{c=1}^{C} \int_{\mathbf{u}_0^c} q(\mathbf{u}_0^c) \log \frac{p(\mathbf{u}_0^c)}{q(\mathbf{u}_0^c)} \mathrm{d}\mathbf{u}_0^c \\ &= -\sum_{c=1}^{C} \text{KL}[q(\mathbf{u}_0^c) || p(\mathbf{u}_0^c)] \end{aligned} \tag{17}$$

The Expected log likelihood (ELL) is given by:

$$\begin{aligned} \text{ELL} &= \int_{\mathbf{f}_0^1} \dots \int_{\mathbf{f}_0^C} \int_{\mathbf{u}_0^1} \dots \int_{\mathbf{u}_0^C} q(\overline{\mathbf{f}}_0, \overline{\mathbf{u}}_0) \log p(\mathbf{y} \mid \overline{\mathbf{f}}_0) \mathrm{d}\mathbf{f}_0^1 \dots \mathrm{d}\mathbf{f}_0^C \mathrm{d}\mathbf{u}_0^1 \dots \mathrm{d}\mathbf{u}_0^C \\ &= \int_{\mathbf{f}_0^1} \dots \int_{\mathbf{f}_0^C} q(\overline{\mathbf{f}}_0) \log \prod_{n=1}^{N} p(y^n \mid \overline{f}_{0,n}) \mathrm{d}\mathbf{f}_0^1 \dots \mathrm{d}\mathbf{f}_0^C \\ &= \int_{\mathbf{f}_0^1} \dots \int_{\mathbf{f}_0^C} q(\mathbf{f}_0^1) \dots q(\mathbf{f}_0^C) \log \prod_{n=1}^{N} \prod_{c=1}^{C} \pi_c(\overline{f}_{0,n})^{\mathbb{I}(y^n=c)} \mathrm{d}\mathbf{f}_0^1 \dots \mathrm{d}\mathbf{f}_0^C \\ &= \sum_{n=1}^{N} \sum_{c=1}^{C} \mathbb{I}(y^n = c) \int_{f_{0,n}^1} \dots \int_{f_{0,n}^C} q(f_{0,n}^1) \dots q(f_{0,n}^C) \log \pi_c(\overline{f}_{0,n}) \mathrm{d}f_{0,n}^1 \dots \mathrm{d}f_{0,n}^C \end{aligned} \tag{18}$$

recovering the bound of the main paper (Eq. 2):

$$\begin{aligned}\text{ELBO} = &\sum_{n=1}^{N}\sum_{c=1}^{C}\mathbb{I}(y^n=c)\int_{f_{0,n}^1}\dots\int_{f_{0,n}^C} q(f_{0,n}^1)\dots q(f_{0,n}^C)\log\pi_c(\overline{f}_{0,n})\mathrm{d}f_{0,n}^1\dots\mathrm{d}f_{0,n}^C \\ &-\sum_{c=1}^{C}\text{KL}[q(\mathbf{u}_0^c)||p(\mathbf{u}_0^c)]\end{aligned} \tag{19}$$

Note that this bound is amenable to stochastic optimization using minibatches, where the integrals are approximated by Monte Carlo using reparameterized gradients (a.k.a. path-wise gradients). The KL can be computed in closed form. Most importantly, each $q(f_{0,n}^c)$ is a univariate Gaussian distribution given by:

$$\begin{aligned}q(f_{0,n}^c) = \mathcal{N}(f_{0,n}^c\,|&K_{\nu\mathbf{x}^n,\mathbf{Z}^c}^c {K_{\nu\mathbf{Z}^c,\mathbf{Z}^c}^c}^{-1}\mathbf{m}^c, \\ &K_{\nu\mathbf{x}^n,\mathbf{x}^n}^c - K_{\nu\mathbf{x}^n,\mathbf{Z}^c}^c {K_{\nu\mathbf{Z}^c,\mathbf{Z}^c}^c}^{-1}[K_{\nu\mathbf{Z}^c,\mathbf{Z}^c}^c + \mathbf{S}^c]{K_{\nu\mathbf{Z}^c,\mathbf{Z}^c}^c}^{-1}K_{\nu\mathbf{Z}^c,\mathbf{x}^n}^c)\end{aligned} \tag{20}$$

obtained by solving $\int_{\mathbf{u}_0^1}\dots\int_{\mathbf{u}_0^C}\prod_{c=1}^C p(\mathbf{f}_0^c \mid \mathbf{u}_0^c,\mathbf{X},\mathbf{Z}^c,\nu_c)q(\mathbf{u}_0^c \mid \mathbf{m}^c,\mathbf{S}^c)\mathrm{d}\mathbf{u}_0^1\dots\mathrm{d}\mathbf{u}_0^C = \prod_{c=1}^C\int_{\mathbf{u}_0^c} p(\mathbf{f}_0^c \mid \mathbf{u}_0^c,\mathbf{X},\mathbf{Z}^c,\nu_c)q(\mathbf{u}_0^c \mid \mathbf{m}^c,\mathbf{S}^c)\mathrm{d}\mathbf{u}_0^c$. This computation needs to be computed C times requiring a complexity of $\mathcal{O}(CM^3 + CNM^2)$, which can be reduced to $\mathcal{O}(M^3 + CNM^2)$ if the inducing points are shared. We can gain additional performance if the kernel is shared as noted in App. C. However, as shown in the experiments sharing kernel and inducing points can drop performance.

A.2. Classification with Efficient Transformed Gaussian Processes

We now present the derivations required for the proposed model. This model is specified by transforming a single sample from a GP using C invertible transformations $\overline{\mathbb{G}}_{\boldsymbol{\theta}}$ by the following generative procedure:

$$\begin{aligned}f_0(\cdot) &\sim \text{GP}(0, K_\nu(\cdot,\cdot)) \\ f_K^1(\cdot) = \mathbb{G}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(f_0(\cdot)),\ \ \cdots\ \ f_K^C(\cdot) &= \mathbb{G}^C_{\boldsymbol{\theta}(\mathbf{W}^C,\mathbf{X})}(f_0(\cdot)).\end{aligned} \tag{21}$$

The prior distribution over C processes is derived as follows. For exemplification purposes consider $C=3$ and consider the processes evaluation at the index set $\mathbf{X}$, then we have:

$$\begin{aligned}&\mathbf{f}_0 \sim p(\mathbf{f}_0 \mid \mathbf{X},\nu) \\ &\mathbf{f}_K^1 = \mathbb{G}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_0);\ \mathbf{f}_K^1 = \mathbb{G}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}\circ\mathbb{H}^2_{\boldsymbol{\theta}(\mathbf{W}^2,\mathbf{X})}(\mathbf{f}_K^2);\ \mathbf{f}_K^1 = \mathbb{G}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}\circ\mathbb{H}^3_{\boldsymbol{\theta}(\mathbf{W}^3,\mathbf{X})}(\mathbf{f}_K^3) \\ &\mathbf{f}_K^2 = \mathbb{G}^2_{\boldsymbol{\theta}(\mathbf{W}^2,\mathbf{X})}(\mathbf{f}_0);\ \mathbf{f}_K^2 = \mathbb{G}^2_{\boldsymbol{\theta}(\mathbf{W}^2,\mathbf{X})}\circ\mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1);\ \mathbf{f}_K^2 = \mathbb{G}^2_{\boldsymbol{\theta}(\mathbf{W}^2,\mathbf{X})}\circ\mathbb{H}^3_{\boldsymbol{\theta}(\mathbf{W}^3,\mathbf{X})}(\mathbf{f}_K^3) \\ &\mathbf{f}_K^3 = \mathbb{G}^3_{\boldsymbol{\theta}(\mathbf{W}^3,\mathbf{X})}(\mathbf{f}_0);\ \mathbf{f}_K^3 = \mathbb{G}^3_{\boldsymbol{\theta}(\mathbf{W}^3,\mathbf{X})}\circ\mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1);\ \mathbf{f}_K^3 = \mathbb{G}^3_{\boldsymbol{\theta}(\mathbf{W}^3,\mathbf{X})}\circ\mathbb{H}^2_{\boldsymbol{\theta}(\mathbf{W}^2,\mathbf{X})}(\mathbf{f}_K^2)\end{aligned} \tag{22}$$

with $\mathbb{H} \coloneqq \mathbb{G}^{-1}$. In order to define the prior probability of the classes we first observe that the following conditional independence holds from the construction introduced above:

$$\begin{aligned}p(\mathbf{f}_K^1,\mathbf{f}_K^2,\mathbf{f}_K^3) &= p(\mathbf{f}_K^1)p(\mathbf{f}_K^2 \mid \mathbf{f}_K^1)p(\mathbf{f}_K^3 \mid \mathbf{f}_K^1,\cancel{\mathbf{f}_K^2}) \\ p(\mathbf{f}_K^2,\mathbf{f}_K^1,\mathbf{f}_K^3) &= p(\mathbf{f}_K^2)p(\mathbf{f}_K^1 \mid \mathbf{f}_K^2)p(\mathbf{f}_K^3 \mid \mathbf{f}_K^2,\cancel{\mathbf{f}_K^1}) \\ p(\mathbf{f}_K^3,\mathbf{f}_K^1,\mathbf{f}_K^2) &= p(\mathbf{f}_K^3)p(\mathbf{f}_K^1 \mid \mathbf{f}_K^3)p(\mathbf{f}_K^2 \mid \mathbf{f}_K^3,\cancel{\mathbf{f}_K^1})\end{aligned} \tag{23}$$

where we have chosen to write the 3 out of 6 possibilities just for exemplification purposes. This conditional independence holds because the probability of $\mathbf{f}_K^3$ given $\mathbf{f}_K^1,\mathbf{f}_K^2$ is given by a direct mapping either from $\mathbf{f}_K^1$ or $\mathbf{f}_K^2$ as illustrated in Eq. 22. We define the *pivot* to be the member on which we *always* condition, i.e. if $p(\mathbf{f}_K^3 \mid \mathbf{f}_K^2,\mathbf{f}_K^1) = p(\mathbf{f}_K^3 \mid \mathbf{f}_K^1)$ then the *pivot* is $\mathbf{f}_K^1$. Note, however, that it will also be valid to choose any other member as a *pivot*, for example $p(\mathbf{f}_K^3 \mid \mathbf{f}_K^2,\mathbf{f}_K^1) = p(\mathbf{f}_K^3 \mid \mathbf{f}_K^2)$. Finally, note that we can write the conditional distribution $p(\mathbf{f}_K^3 \mid \mathbf{f}_K^1)$ as:

$$p(\mathbf{f}_K^3 \mid \mathbf{f}_K^1) = \delta\left(\mathbf{f}_K^3 - \mathbb{G}^3_{\boldsymbol{\theta}(\mathbf{W}^3,\mathbf{X})}\circ\mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1)\right) \tag{24}$$

with δ being the Dirac measure. Using both observations we can write the prior joint conditional distribution over the classes as:

$$p(\overline{\mathbf{f}}_K \mid \overline{\mathbb{G}}_{\boldsymbol{\theta}}, \overline{\mathbf{W}}, \mathbf{X}, \nu) = p(\mathbf{f}_0 \mid \mathbf{X}, \nu) \prod_{k=0}^{K^1-1} \left| \det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}^1_K)}{\partial \mathbf{f}^1_K} \right|^{-1} \prod_{c=2}^{C} \delta\left(\mathbf{f}^c_K - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c,\mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}^1_K)\right) \tag{25}$$

recovering the expression in Eq. 6 in the main paper. The overall joint is given by:

$$p(\overline{\mathbf{f}}_K, \overline{\mathbf{W}} \mid \overline{\mathbb{G}}_{\boldsymbol{\theta}}, \overline{\lambda}, \mathbf{X}, \nu) = p(\overline{\mathbf{f}}_K \mid \overline{\mathbb{G}}_{\boldsymbol{\theta}}, \overline{\mathbf{W}}, \mathbf{X}, \nu) p(\overline{\mathbf{W}} \mid \overline{\lambda}) \tag{26}$$

with $p(\overline{\mathbf{W}} \mid \overline{\lambda})$ denoting the prior over the parameters of the Bayesian Neural Network (BNN).

A.2.1. Prior Conditional Distribution $p(\overline{\mathbf{f}}_K \mid \overline{\mathbf{u}}_K)$

We now derive the prior conditional distribution $p(\overline{\mathbf{f}}_K \mid \overline{\mathbf{u}}_K)$. In a similar vein to GPs we will derive a sparse variational inference algorithm, from where inducing points need to be incorporated. Note that since we use diagonal flows, the resulting joint distribution is consistent (i.e. is a finite dimensional realization of a stochastic process), which means we can extend its index set introducing inducing points $\overline{\mathbf{u}}_K$ at inducing locations $\mathbf{Z}$, similar to what we do in GPs.

First, note that following the previous section we can write the marginal distribution at the inducing points by:

$$p(\overline{\mathbf{u}}_K \mid \overline{\mathbb{G}}_{\boldsymbol{\theta}}, \overline{\mathbf{W}}, \mathbf{Z}, \nu) = p(\mathbf{u}_0 \mid \mathbf{Z}, \nu) \prod_{k=0}^{K^1-1} \left| \det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1,\mathbf{Z})}(\mathbf{u}^1_K)}{\partial \mathbf{u}^1_K} \right|^{-1} \prod_{c=2}^{C} \delta\left(\mathbf{u}^c_K - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c,\mathbf{Z})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{Z})}(\mathbf{u}^1_K)\right). \tag{27}$$

The overall joint can be derived following a similar procedure. Note that we have:

$$\begin{aligned}
&\mathbf{f}_0, \mathbf{u}_0 \sim p(\mathbf{f}_0, \mathbf{u}_0 \mid \mathbf{X}, \mathbf{Z}, \lambda) \\
&\mathbf{f}^1_K = \mathbb{G}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_0); \quad \mathbf{f}^1_K = \mathbb{G}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})} \circ \mathbb{H}^2_{\boldsymbol{\theta}(\mathbf{W}^2,\mathbf{X})}(\mathbf{f}^2_K); \quad \mathbf{f}^1_K = \mathbb{G}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})} \circ \mathbb{H}^3_{\boldsymbol{\theta}(\mathbf{W}^3,\mathbf{X})}(\mathbf{f}^3_K) \\
&\mathbf{u}^1_K = \mathbb{G}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{Z})}(\mathbf{u}_0); \quad \mathbf{u}^1_K = \mathbb{G}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{Z})} \circ \mathbb{H}^2_{\boldsymbol{\theta}(\mathbf{W}^2,\mathbf{Z})}(\mathbf{u}^2_K); \quad \mathbf{u}^1_K = \mathbb{G}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{Z})} \circ \mathbb{H}^3_{\boldsymbol{\theta}(\mathbf{W}^3,\mathbf{Z})}(\mathbf{u}^3_K) \\
&\mathbf{f}^2_K = \mathbb{G}^2_{\boldsymbol{\theta}(\mathbf{W}^2,\mathbf{X})}(\mathbf{f}_0); \quad \mathbf{f}^2_K = \mathbb{G}^2_{\boldsymbol{\theta}(\mathbf{W}^2,\mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}^1_K); \quad \mathbf{f}^2_K = \mathbb{G}^2_{\boldsymbol{\theta}(\mathbf{W}^2,\mathbf{X})} \circ \mathbb{H}^3_{\boldsymbol{\theta}(\mathbf{W}^3,\mathbf{X})}(\mathbf{f}^3_K) \\
&\mathbf{u}^2_K = \mathbb{G}^2_{\boldsymbol{\theta}(\mathbf{W}^2,\mathbf{Z})}(\mathbf{u}_0); \quad \mathbf{u}^2_K = \mathbb{G}^2_{\boldsymbol{\theta}(\mathbf{W}^2,\mathbf{Z})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{Z})}(\mathbf{u}^1_K); \quad \mathbf{u}^2_K = \mathbb{G}^2_{\boldsymbol{\theta}(\mathbf{W}^2,\mathbf{Z})} \circ \mathbb{H}^3_{\boldsymbol{\theta}(\mathbf{W}^3,\mathbf{Z})}(\mathbf{u}^3_K) \\
&\mathbf{f}^3_K = \mathbb{G}^3_{\boldsymbol{\theta}(\mathbf{W}^3,\mathbf{X})}(\mathbf{f}_0); \quad \mathbf{f}^3_K = \mathbb{G}^3_{\boldsymbol{\theta}(\mathbf{W}^3,\mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}^1_K); \quad \mathbf{f}^3_K = \mathbb{G}^3_{\boldsymbol{\theta}(\mathbf{W}^3,\mathbf{X})} \circ \mathbb{H}^2_{\boldsymbol{\theta}(\mathbf{W}^2,\mathbf{X})}(\mathbf{f}^2_K) \\
&\mathbf{u}^3_K = \mathbb{G}^3_{\boldsymbol{\theta}(\mathbf{W}^3,\mathbf{Z})}(\mathbf{u}_0); \quad \mathbf{u}^3_K = \mathbb{G}^3_{\boldsymbol{\theta}(\mathbf{W}^3,\mathbf{Z})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{Z})}(\mathbf{u}^1_K); \quad \mathbf{u}^3_K = \mathbb{G}^3_{\boldsymbol{\theta}(\mathbf{W}^3,\mathbf{Z})} \circ \mathbb{H}^2_{\boldsymbol{\theta}(\mathbf{W}^2,\mathbf{Z})}(\mathbf{u}^2_K)
\end{aligned}$$

Following similar ideas as before, the pivots are now defined to be $\mathbf{f}^1_K$ and $\mathbf{u}^1_K$. We can also apply a similar conditional independence, and note that the joint distribution over the non *pivots* $\mathbf{f}_K, \mathbf{u}_K$ also factorizes. Conditional independence holds because any $\mathbf{f}^c_K, \mathbf{u}^c_K$ only depends on $\mathbf{f}^1_K, \mathbf{u}^1_K$ by a direct mapping; and the conditional distribution over the non *pivots*

$p(\mathbf{f}_K^c, \mathbf{u}_K^c \mid \mathbf{f}_K^1, \mathbf{u}_K^1) = p(\mathbf{f}_K^c \mid \mathbf{f}_K^1) p(\mathbf{u}_K^c \mid \mathbf{u}_K^1)$ factorizes since $\mathbf{f}_K^c$ only depends on $\mathbf{f}_K^1$ and $\mathbf{u}_K^c$ on $\mathbf{u}_K^1$. Writing:

$$
\begin{aligned}
&p(\mathbf{f}_K^1, \mathbf{u}_K^1, \mathbf{f}_K^2, \mathbf{u}_K^2, \mathbf{f}_K^3, \mathbf{u}_K^3) = \\
&p(\mathbf{f}_K^1, \mathbf{u}_K^1) p(\mathbf{f}_K^2, \mathbf{u}_K^2 \mid \mathbf{f}_K^1, \mathbf{u}_K^1) p(\mathbf{f}_K^3, \mathbf{u}_K^3 \mid \mathbf{f}_K^1, \mathbf{u}_K^1, \cancel{\mathbf{f}_K^2, \mathbf{u}_K^2}) = \\
&p(\mathbf{f}_K^1, \mathbf{u}_K^1) p(\mathbf{f}_K^2 \mid \mathbf{f}_K^1) p(\mathbf{u}_K^2 \mid \mathbf{u}_K^1) p(\mathbf{f}_K^3 \mid \mathbf{f}_K^1) p(\mathbf{u}_K^3 \mid \mathbf{u}_K^1) = \\
&\underbrace{\underbrace{p(\mathbf{f}_0 \mid \mathbf{u}_0) \prod_{k=0}^{K^1-1} \left| \det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1, \mathbf{X})}(\mathbf{f}_K^1)}{\partial \mathbf{f}_K^1} \right|^{-1}}_{p(\mathbf{f}_K^1 | \mathbf{u}_K^1)} \underbrace{p(\mathbf{u}_0) \prod_{k=0}^{K^1-1} \left| \det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1, \mathbf{Z})}(\mathbf{u}_K^1)}{\partial \mathbf{u}_K^1} \right|^{-1}}_{p(\mathbf{u}_K^1)}}_{p(\mathbf{f}_K^1, \mathbf{u}_K^1)} \\
&\underbrace{\delta\left(\mathbf{f}_K^2 - \mathbb{G}^2_{\boldsymbol{\theta}(\mathbf{W}^2, \mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1, \mathbf{X})}(\mathbf{f}_K^1)\right) \delta\left(\mathbf{u}_K^2 - \mathbb{G}^2_{\boldsymbol{\theta}(\mathbf{W}^2, \mathbf{Z})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1, \mathbf{Z})}(\mathbf{u}_K^1)\right)}_{p(\mathbf{f}_K^2, \mathbf{u}_K^2 | \mathbf{f}_K^1, \mathbf{u}_K^1) = p(\mathbf{f}_K^2 | \mathbf{f}_K^1) p(\mathbf{u}_K^2 | \mathbf{u}_K^1)} \\
&\underbrace{\delta\left(\mathbf{f}_K^3 - \mathbb{G}^3_{\boldsymbol{\theta}(\mathbf{W}^3, \mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1, \mathbf{X})}(\mathbf{f}_K^1)\right) \delta\left(\mathbf{u}_K^3 - \mathbb{G}^3_{\boldsymbol{\theta}(\mathbf{W}^3, \mathbf{Z})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1, \mathbf{Z})}(\mathbf{u}_K^1)\right)}_{p(\mathbf{f}_K^3, \mathbf{u}_K^3 | \mathbf{f}_K^1, \mathbf{u}_K^1) = p(\mathbf{f}_K^3 | \mathbf{f}_K^1) p(\mathbf{u}_K^3 | \mathbf{u}_K^1)}
\end{aligned}
\tag{28}
$$

Because we use a diagonal flow, the full Jacobian factorizes as $\prod_{k=0}^{K^1-1} \left| \det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1, \mathbf{X})}(\mathbf{f}_K^1)}{\partial \mathbf{f}_K^1} \right|^{-1} \prod_{k=0}^{K^1-1} \left| \det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1, \mathbf{Z})}(\mathbf{u}_K^1)}{\partial \mathbf{u}_K^1} \right|^{-1}$, allowing us to explicitly write $p(\mathbf{f}_K^1 \mid \mathbf{u}_K^1)$ and $p(\mathbf{u}_K^1)$ (see appendix of Maroñas et al. (2021)). Thus, the overall joint distribution is given by:

$$
\begin{aligned}
&p(\overline{\mathbf{f}}_K, \overline{\mathbf{u}}_K, \overline{\mathbf{W}} \mid \overline{\mathbb{G}}_{\boldsymbol{\theta}}, \mathbf{X}, \mathbf{Z}, \overline{\lambda}, \nu) = \prod_{c=1}^{C} p(\mathbf{W}^c \mid \lambda^c) \\
&\underbrace{p(\mathbf{f}_0 \mid \mathbf{u}_0, \mathbf{X}, \mathbf{Z}, \nu) \prod_{k=0}^{K^1-1} \left| \det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1, \mathbf{X})}(\mathbf{f}_k^1)}{\partial \mathbf{f}_k^1} \right|^{-1} \prod_{c=2}^{C} \delta\left(\mathbf{f}_K^c - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c, \mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1, \mathbf{X})}(\mathbf{f}_K^1)\right)}_{p(\overline{\mathbf{f}}_K | \overline{\mathbf{u}}_K)} \\
&\underbrace{p(\mathbf{u}_0 \mid \mathbf{Z}, \nu) \prod_{k=0}^{K^1-1} \left| \det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1, \mathbf{Z})}(\mathbf{u}_k^1)}{\partial \mathbf{u}_k^1} \right|^{-1} \prod_{c=2}^{C} \delta\left(\mathbf{u}_K^c - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c, \mathbf{Z})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1, \mathbf{Z})}(\mathbf{u}_K^1)\right)}_{p(\overline{\mathbf{u}}_K)}
\end{aligned}
\tag{29}
$$

where now the *pivots* are $\mathbf{u}_K^1$ and $\mathbf{f}_K^1$. To fully characterize the joint distribution we shall derive where the expression for $p(\overline{\mathbf{f}}_K \mid \overline{\mathbf{u}}_K)$ in Eq. 29 comes from. This conditional distribution is derived by inspection as follows. We factorize the joint distribution in the following two equivalent ways:

$$
\begin{aligned}
&p(\mathbf{f}_K^1, \mathbf{f}_K^2, \mathbf{f}_K^3, \mathbf{u}_K^1, \mathbf{u}_K^2, \mathbf{u}_K^3) = \\
&p(\mathbf{f}_K^1, \mathbf{f}_K^2, \mathbf{f}_K^3 \mid \mathbf{u}_K^1, \mathbf{u}_K^2, \mathbf{u}_K^3) p(\mathbf{u}_K^1, \mathbf{u}_K^2, \mathbf{u}_K^3) = \\
&p(\mathbf{f}_K^1, \mathbf{u}_K^1) p(\mathbf{f}_K^2, \mathbf{u}_K^2 \mid \mathbf{u}_K^1, \mathbf{u}_K^1) p(\mathbf{f}_K^3, \mathbf{u}_K^3 \mid \mathbf{f}_K^1, \mathbf{u}_K^1)
\end{aligned}
$$

and used the form of the third line, which is the one we know how to write (Eq. 28), to derive the expression for the second

line, which is the object of our interest. The expression for the third line has already been written and is given by:

$$
\begin{aligned}
&p(\mathbf{f}_K^1, \mathbf{f}_K^2, \mathbf{f}_K^3, \mathbf{u}_K^1, \mathbf{u}_K^2, \mathbf{u}_K^3) = \\
&\underbrace{p(\mathbf{f}_0 \mid \mathbf{u}_0) p(\mathbf{u}_0) \prod_{k=0}^{K^1-1} \left| \det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1)}{\partial \mathbf{f}_K^1} \right|^{-1} \prod_{k=0}^{K^1-1} \left| \det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1,\mathbf{Z})}(\mathbf{u}_K^1)}{\partial \mathbf{u}_K^1} \right|^{-1}}_{p(\mathbf{f}_K^1, \mathbf{u}_K^1)} \\
&\underbrace{\delta\left(\mathbf{f}_K^2 - \mathbb{G}^2_{\boldsymbol{\theta}(\mathbf{W}^2,\mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1)\right) \delta\left(\mathbf{u}_K^2 - \mathbb{G}^2_{\boldsymbol{\theta}(\mathbf{W}^2,\mathbf{Z})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{Z})}(\mathbf{u}_K^1)\right)}_{p(\mathbf{f}_K^2, \mathbf{u}_K^2 \mid \mathbf{u}_K^1, \mathbf{u}_K^1)} \\
&\underbrace{\delta\left(\mathbf{f}_K^3 - \mathbb{G}^3_{\boldsymbol{\theta}(\mathbf{W}^3,\mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1)\right) \delta\left(\mathbf{u}_K^3 - \mathbb{G}^3_{\boldsymbol{\theta}(\mathbf{W}^3,\mathbf{Z})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{Z})}(\mathbf{u}_K^1)\right)}_{p(\mathbf{f}_K^3, \mathbf{u}_K^3 \mid \mathbf{f}_K^1, \mathbf{u}_K^1)}
\end{aligned}
\tag{30}
$$

Then, since we know the form of $p(\mathbf{u}_K^1, \mathbf{u}_K^2, \mathbf{u}_K^3)$:

$$
\begin{aligned}
&p(\mathbf{u}_K^1, \mathbf{u}_K^2, \mathbf{u}_K^3) = \\
&p(\mathbf{u}_0) \prod_{k=0}^{K^1-1} \left| \det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1,\mathbf{Z})}(\mathbf{u}_K^1)}{\partial \mathbf{u}_K^1} \right|^{-1} \\
&\delta\left(\mathbf{u}_K^2 - \mathbb{G}^2_{\boldsymbol{\theta}} \circ \mathbb{H}^1_{\boldsymbol{\theta}}(\mathbf{u}_K^1)\right) \delta\left(\mathbf{u}_K^3 - \mathbb{G}^3_{\boldsymbol{\theta}} \circ \mathbb{H}^1_{\boldsymbol{\theta}}(\mathbf{u}_K^1)\right)
\end{aligned}
\tag{31}
$$

then, by careful inspection of Eq. 30 we can derive the conditional distribution, which is given by:

$$
\begin{aligned}
&p(\mathbf{f}_K^1, \mathbf{f}_K^2, \mathbf{f}_K^3 \mid \mathbf{u}_K^1, \mathbf{u}_K^2, \mathbf{u}_K^3) = \\
&p(\mathbf{f}_0 \mid \mathbf{u}_0) \prod_{k=0}^{K^1-1} \left| \det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1)}{\partial \mathbf{f}_K^1} \right|^{-1} \\
&\delta\left(\mathbf{f}_K^2 - \mathbb{G}^2_{\boldsymbol{\theta}(\mathbf{W}^2,\mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1)\right) \delta\left(\mathbf{f}_K^3 - \mathbb{G}^3_{\boldsymbol{\theta}(\mathbf{W}^3,\mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1)\right)
\end{aligned}
\tag{32}
$$

where we have just seen (marked in red in Eq. 30) which elements from the full joint belong to the marginal $p(\mathbf{u}_K^1, \mathbf{u}_K^2, \mathbf{u}_K^3)$, and thus the remaining must belong to the conditional. This gives the prior conditional for C processes:

$$
\begin{aligned}
p(\overline{\mathbf{f}}_K \mid \overline{\mathbf{u}}_K, \overline{\mathbf{W}}, \mathbf{X}, \mathbf{Z}, \nu) = &\underbrace{p(\mathbf{f}_0 \mid \mathbf{u}_0, \mathbf{X}, \mathbf{Z}, \nu) \prod_{k=0}^{K^1-1} \left| \det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1)}{\partial \mathbf{f}_K^1} \right|^{-1}}_{p(\mathbf{f}_K^1 \mid \mathbf{u}_K^1)} \\
&\prod_{c=2}^{C} \delta\left(\mathbf{f}_K^c - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c,\mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1)\right)
\end{aligned}
\tag{33}
$$

matching the result in Eq. 29. With this, we are now ready to derive the sparse variational inference algorithm.

A.2.2. Marginal Variational Distribution $q(\overline{\mathbf{f}}_K)$

The variational distribution is defined following the ideas from Maroñas et al. (2021); Titsias (2009); Hensman et al. (2013):

$$
q(\overline{\mathbf{f}}_K, \overline{\mathbf{u}}_K, \overline{\mathbf{W}}) = p(\overline{\mathbf{f}}_K \mid \overline{\mathbf{u}}_K, \overline{\mathbf{W}}) q(\overline{\mathbf{u}}_K \mid \overline{\mathbf{W}}) q(\overline{\mathbf{W}})
\tag{34}
$$

where we use the conditional model's prior derived in the previous section and a marginal conditional variational distribution that is defined by warping a multivariate Gaussian in the original GP space $q(\mathbf{u}_0 \mid \mathbf{m}, \mathbf{S})$ using $\overline{\mathbb{G}}$, where $\mathbf{m} \in \mathbb{R}^M$, $\mathbf{S} \in \mathbb{R}^{M \times M}$, with the flows from the prior $\overline{\mathbb{G}}_{\boldsymbol{\theta}}$:

$$
\begin{aligned}
q(\overline{\mathbf{u}}_K \mid \mathbf{m}, \mathbf{S}, \overline{\mathbf{W}}, \mathbf{Z}) = q(\mathbf{u}_0 \mid \mathbf{m}, \mathbf{S}) &\prod_{k=0}^{K^1-1} \left| \det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1,\mathbf{Z})}(\mathbf{u}_k^1)}{\partial \mathbf{u}_k^1} \right|^{-1} \\
&\prod_{c=2}^{C} \delta\left(\mathbf{u}_K^c - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c,\mathbf{Z})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{Z})}(\mathbf{u}_K^1)\right)
\end{aligned}
\tag{35}
$$

and where the distribution over the NN weights factorizes:

$$q(\overline{\mathbf{W}} \mid \overline{\phi}) = \prod_{c=1}^{C} q(\mathbf{W}^c \mid \phi_c) \tag{36}$$

where $\overline{\phi}$ denote variational parameters. Note that the dependence of the marginal $q(\overline{\mathbf{u}}_K)$ on $\overline{\mathbf{W}}$ is required since this distribution is parameterized by the flows of the prior and so inference over $\overline{\mathbf{W}}$ requires dependence between $q(\overline{\mathbf{u}}_K)$ and $q(\overline{\mathbf{W}} \mid \overline{\phi})$.

To derive our inference algorithm, we need to show how to integrate out inducing points, which turns out that can be done analytically when using diagonal flows, as in Maroñas et al. (2021):

$$\begin{aligned}
q(\overline{\mathbf{f}}_K \mid \overline{\mathbf{W}}) &= \int_{\mathbf{u}_K^1} \dots \int_{\mathbf{u}_K^C} p(\overline{\mathbf{f}}_K \mid \overline{\mathbf{u}}_K) q(\overline{\mathbf{u}}_K) \mathrm{d}\mathbf{u}_K^1 \dots \mathrm{d}\mathbf{u}_K^C \\
&= \int_{\mathbf{u}_K^1} \dots \int_{\mathbf{u}_K^C} p(\mathbf{f}_K^1 \mid \mathbf{u}_K^1) \prod_{c=2}^{C} \delta\left(\mathbf{f}_K^c - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c,\mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1)\right) \\
&\quad q(\mathbf{u}_K^1) \prod_{c=2}^{C} \delta\left(\mathbf{u}_K^c - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c,\mathbf{Z})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{Z})}(\mathbf{u}_K^1)\right) \mathrm{d}\mathbf{u}_K^1 \dots \mathrm{d}\mathbf{u}_K^C \\
&= \prod_{c=2}^{C} \delta\left(\mathbf{f}_K^c - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c,\mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1)\right) \\
&\quad \int_{\mathbf{u}_K^1} \dots \int_{\mathbf{u}_K^C} p(\mathbf{f}_K^1 \mid \mathbf{u}_K^1) q(\mathbf{u}_K^1) \prod_{c=2}^{C} \delta\left(\mathbf{u}_K^c - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c,\mathbf{Z})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{Z})}(\mathbf{u}_K^1)\right) \mathrm{d}\mathbf{u}_K^1 \dots \mathrm{d}\mathbf{u}_K^C \\
&= \prod_{c=2}^{C} \delta\left(\mathbf{f}_K^c - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c,\mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1)\right) \int_{\mathbf{u}_K^1} p(\mathbf{f}_K^1 \mid \mathbf{u}_K^1) q(\mathbf{u}_K^1) \mathrm{d}\mathbf{u}_K^1 \\
&= \prod_{c=2}^{C} \delta\left(\mathbf{f}_K^c - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c,\mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1)\right) \prod_{k=0}^{K^1-1} \left| \det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1)}{\partial \mathbf{f}_K^1} \right|^{-1} \\
&\quad \int p(\mathbf{f}_0 \mid \mathbf{u}_0) q(\mathbf{u}_0) \mathrm{d}\mathbf{u}_0 \\
&= q(\mathbf{f}_0) \prod_{k=0}^{K^1-1} \left| \det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1)}{\partial \mathbf{f}_K^1} \right|^{-1} \prod_{c=2}^{C} \delta\left(\mathbf{f}_K^c - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c,\mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1)\right)
\end{aligned} \tag{37}$$

where $q(\mathbf{f}_0)$ is given by Eq. 20. Note that the form of this marginal variational distribution $q(\overline{\mathbf{f}}_K)$ implies the following generative procedure already used in the definition of the ETGP:

$$\begin{gathered}
\mathbf{f}_0 \sim q(\mathbf{f}_0) \\
\mathbf{f}_K^1 = \mathbb{G}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_0),\ \mathbf{f}_K^2 = \mathbb{G}^2_{\boldsymbol{\theta}(\mathbf{W}^2,\mathbf{X})}(\mathbf{f}_0)\ \ \dots\ \ \mathbf{f}_K^C = \mathbb{G}^C_{\boldsymbol{\theta}(\mathbf{W}^C,\mathbf{X})}(\mathbf{f}_0)
\end{gathered} \tag{38}$$

The sequence of steps used in the derivation are the followings. We start by writing the marginalization in terms of the conditional distribution $p(\overline{\mathbf{f}}_K \mid \overline{\mathbf{u}}_K, \overline{\mathbf{W}})$ and the proposed marginal variational $q(\overline{\mathbf{u}}_K \mid \overline{\mathbf{W}})$. From second to third equality we take out from the integral the terms that do not depend on $\overline{\mathbf{u}}_K$. Then from third to fourth equality we integrate out all $\overline{\mathbf{u}}_K$ except the *pivot* $\mathbf{u}_K^1$. Note that integration here is straightforward since the Dirac measure integrates to 1[5]. This let us with one integral over $\mathbf{u}_K^1$. From fourth to fifth equality, we write $p(\mathbf{f}_K^1 \mid \mathbf{u}_K^1)$ using the expression in Eq. 33, and since the Jacobian does not depend on $\mathbf{u}_K^1$ it is taken out from the integral. Lastly, we apply the LOTUS rule (see appendix in Maroñas et al. (2021) and below) by noting an expectation over $q(\mathbf{u}_K^1)$, which give us a simple Gaussian integral, from which analytical solution $q(\mathbf{f}_0)$ is well known. This distribution coincides with the SVGP marginal variational given by Eq. 20, as in TGPS.

For self-contained purposes we copy the LOTUS rule definition in Maroñas et al. (2021):

[5]Readers concerned with the integration of the Dirac measure in this context can replace it by a Gaussian density taking the limit of $\sigma \to 0$.

LOTUS rule: Given an invertible transformation $\mathbb{G}$, and the distribution $p(\mathbf{f}_K)$ induced by transforming samples from a base distribution $p(\mathbf{f}_0)$, then it holds that expectations of any function $h()$ under $p(\mathbf{f}_K)$ can be computed by integrating w.r.t. the base distribution $p(\mathbf{f}_0)$. This is formally known as probability under change of measure. Formally, the above statement implies:

$$\mathbb{E}_{p(\mathbf{f}_K)}\left[h(\mathbf{f}_K)\right] = \mathbb{E}_{p(\mathbf{f}_0)}\left[h(\mathbb{G}(\mathbf{f}_0))\right] \tag{39}$$

A.2.3. EVIDENCE LOWER BOUND ELBO

The Evidence Lower Bound resulting from the prior model and the variational approximate posterior can be written down as:

$$\begin{aligned}
\text{ELBO} &= \int_{\bar{\mathbf{f}}_K}\int_{\bar{\mathbf{u}}_K}\int_{\overline{\mathbf{W}}} q(\bar{\mathbf{f}}_K, \bar{\mathbf{u}}_K \mid \overline{\mathbf{W}}) q(\overline{\mathbf{W}}) \frac{\log p(\mathbf{y} \mid \bar{\mathbf{f}}_K) p(\bar{\mathbf{f}}_K, \bar{\mathbf{u}}_K \mid \overline{\mathbf{W}}) p(\overline{\mathbf{W}})}{q(\bar{\mathbf{f}}_K, \bar{\mathbf{u}}_K \mid \overline{\mathbf{W}}) q(\overline{\mathbf{W}})} \mathrm{d}\bar{\mathbf{f}}_K \mathrm{d}\bar{\mathbf{u}}_K \mathrm{d}\overline{\mathbf{W}} \\
&= \underbrace{\int_{\bar{\mathbf{f}}_K}\int_{\bar{\mathbf{u}}_K}\int_{\overline{\mathbf{W}}} q(\bar{\mathbf{f}}_K, \bar{\mathbf{u}}_K \mid \overline{\mathbf{W}}) q(\overline{\mathbf{W}}) \log p(\mathbf{y} \mid \bar{\mathbf{f}}_K) \mathrm{d}\bar{\mathbf{f}}_K \mathrm{d}\bar{\mathbf{u}}_K \mathrm{d}\overline{\mathbf{W}}}_{\text{ELL}} \\
&\underbrace{+ \int_{\bar{\mathbf{f}}_K}\int_{\bar{\mathbf{u}}_K}\int_{\overline{\mathbf{W}}} q(\bar{\mathbf{f}}_K, \bar{\mathbf{u}}_K \mid \overline{\mathbf{W}}) q(\overline{\mathbf{W}}) \log \frac{p(\bar{\mathbf{f}}_K, \bar{\mathbf{u}}_K \mid \overline{\mathbf{W}}) p(\overline{\mathbf{W}})}{q(\bar{\mathbf{f}}_K, \bar{\mathbf{u}}_K \mid \overline{\mathbf{W}}) q(\overline{\mathbf{W}})} \mathrm{d}\bar{\mathbf{f}}_K \mathrm{d}\bar{\mathbf{u}}_K \mathrm{d}\overline{\mathbf{W}}}_{-\text{KL}}
\end{aligned} \tag{40}$$

where we again drop the conditioning set, except $\overline{\mathbf{W}}$, for clarity. Working each term separately yields:

$$\begin{aligned}
-\text{KL} &= \int_{\bar{\mathbf{f}}_K}\int_{\bar{\mathbf{u}}_K}\int_{\overline{\mathbf{W}}} q(\bar{\mathbf{f}}_K, \bar{\mathbf{u}}_K \mid \overline{\mathbf{W}}) q(\overline{\mathbf{W}}) \log \frac{p(\bar{\mathbf{f}}_K, \bar{\mathbf{u}}_K \mid \overline{\mathbf{W}}) p(\overline{\mathbf{W}})}{q(\bar{\mathbf{f}}_K, \bar{\mathbf{u}}_K \mid \overline{\mathbf{W}}) q(\overline{\mathbf{W}})} \mathrm{d}\bar{\mathbf{f}}_K \mathrm{d}\bar{\mathbf{u}}_K \mathrm{d}\overline{\mathbf{W}} \\
&= \int_{\bar{\mathbf{f}}_K}\int_{\bar{\mathbf{u}}_K}\int_{\overline{\mathbf{W}}} q(\bar{\mathbf{f}}_K, \bar{\mathbf{u}}_K \mid \overline{\mathbf{W}}) q(\overline{\mathbf{W}}) \log \frac{\cancel{p(\bar{\mathbf{f}}_K \mid \bar{\mathbf{u}}_K, \overline{\mathbf{W}})} p(\bar{\mathbf{u}}_K \mid \overline{\mathbf{W}}) p(\overline{\mathbf{W}})}{\cancel{p(\bar{\mathbf{f}}_K \mid \bar{\mathbf{u}}_K, \overline{\mathbf{W}})} q(\bar{\mathbf{u}}_K \mid \overline{\mathbf{W}}) q(\overline{\mathbf{W}})} \mathrm{d}\bar{\mathbf{f}}_K \mathrm{d}\bar{\mathbf{u}}_K \mathrm{d}\overline{\mathbf{W}} \\
&= \int_{\bar{\mathbf{f}}_K}\int_{\bar{\mathbf{u}}_K}\int_{\overline{\mathbf{W}}} q(\bar{\mathbf{f}}_K, \bar{\mathbf{u}}_K \mid \overline{\mathbf{W}}) q(\overline{\mathbf{W}}) \log \frac{\cancel{\prod_{c=2}^{C} \delta\left(\mathbf{u}_K^c - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c, \mathbf{Z})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1, \mathbf{Z})}(\mathbf{u}_K^1)\right)} p(\bar{\mathbf{u}}_K \mid \overline{\mathbf{W}}) p(\overline{\mathbf{W}})}{\cancel{\prod_{c=2}^{C} \delta\left(\mathbf{u}_K^c - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c, \mathbf{Z})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1, \mathbf{Z})}(\mathbf{u}_K^1)\right)} q(\bar{\mathbf{u}}_K \mid \overline{\mathbf{W}}) q(\overline{\mathbf{W}})} \mathrm{d}\bar{\mathbf{f}}_K \mathrm{d}\bar{\mathbf{u}}_K \mathrm{d}\overline{\mathbf{W}} \\
&+ \int_{\bar{\mathbf{f}}_K}\int_{\bar{\mathbf{u}}_K}\int_{\overline{\mathbf{W}}} q(\bar{\mathbf{f}}_K, \bar{\mathbf{u}}_K \mid \overline{\mathbf{W}}) q(\overline{\mathbf{W}}) \log \frac{\cancel{p(\mathbf{u}_K^1 \mid \mathbf{W}^1)} p(\overline{\mathbf{W}})}{\cancel{q(\mathbf{u}_K^1 \mid \mathbf{W}^1)} q(\overline{\mathbf{W}})} \mathrm{d}\bar{\mathbf{f}}_K \mathrm{d}\bar{\mathbf{u}}_K \mathrm{d}\overline{\mathbf{W}} \\
&= \int_{\bar{\mathbf{u}}_K}\int_{\overline{\mathbf{W}}} q(\mathbf{u}_K^1 \mid \mathbf{W}^1) \prod_{c=2}^{C} \delta\left(\mathbf{u}_K^c - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c, \mathbf{Z})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1, \mathbf{Z})}(\mathbf{u}_K^1)\right) q(\overline{\mathbf{W}}) \log \frac{p(\mathbf{u}_K^1 \mid \mathbf{W}^1) p(\overline{\mathbf{W}})}{q(\mathbf{u}_K^1 \mid \mathbf{W}^1) q(\overline{\mathbf{W}})} \mathrm{d}\bar{\mathbf{u}}_K \mathrm{d}\overline{\mathbf{W}} \\
&= \int_{\mathbf{u}_K^1}\int_{\overline{\mathbf{W}}} q(\mathbf{u}_K^1 \mid \mathbf{W}^1) q(\overline{\mathbf{W}}) \log \frac{p(\mathbf{u}_K^1 \mid \mathbf{W}^1) p(\overline{\mathbf{W}})}{q(\mathbf{u}_K^1 \mid \mathbf{W}^1) q(\overline{\mathbf{W}})} \mathrm{d}\mathbf{u}_K^1 \mathrm{d}\overline{\mathbf{W}} \\
&= \int_{\overline{\mathbf{W}}} q(\overline{\mathbf{W}}) \int_{\mathbf{u}_K^1} q(\mathbf{u}_K^1 \mid \mathbf{W}^1) \log \frac{p(\mathbf{u}_K^1 \mid \mathbf{W}^1)}{q(\mathbf{u}_K^1 \mid \mathbf{W}^1)} \mathrm{d}\mathbf{u}_K^1 \mathrm{d}\overline{\mathbf{W}} + \int_{\overline{\mathbf{W}}} q(\overline{\mathbf{W}}) \log \frac{p(\overline{\mathbf{W}})}{q(\overline{\mathbf{W}})} \mathrm{d}\overline{\mathbf{W}} \\
&= \int_{\overline{\mathbf{W}}} q(\overline{\mathbf{W}}) \int_{\mathbf{u}_0} q(\mathbf{u}_0) \log \frac{p(\mathbf{u}_0)}{q(\mathbf{u}_0)} \mathrm{d}\mathbf{u}_0 \mathrm{d}\overline{\mathbf{W}} + \int_{\overline{\mathbf{W}}} q(\overline{\mathbf{W}}) \log \frac{p(\overline{\mathbf{W}})}{q(\overline{\mathbf{W}})} \mathrm{d}\overline{\mathbf{W}} \\
&= -\text{KL}[q(\mathbf{u}_0) || p(\mathbf{u}_0)] - \text{KL}[q(\overline{\mathbf{W}}) || p(\overline{\mathbf{W}})]
\end{aligned} \tag{41}$$

where in the second and third equalities we cancel common terms. From equality 3 to 4 we integrate out all $\bar{\mathbf{f}}_K$ since nothing depends on them. In step from equality 4 to 5 we integrate out the Dirac measures over all the non-*pivot* elements. From equality 5 to 6 we separate expectations and step 6 to 7 can be derived in two ways. First, since KL is invariant under a parameter transformation (reparameterization) and both the prior and variational distributions are transformed with the same warping function $\overline{\mathbb{G}}_{\boldsymbol{\theta}}$, then the KL can be written as that on the original GP space. Another way to derive this KL is by noting an expected value of a log-ratio w.r.t. $q(\bar{\mathbf{u}}_K)$, allowing us to apply the LOTUS rule, and corresponding Jacobian cancellations.

More precisely:

$$
\begin{aligned}
&\int_{\mathbf{u}_K^1} q(\mathbf{u}_K^1) \log \frac{p(\mathbf{u}_K^1)}{q(\mathbf{u}_K^1)} \mathrm{d}\mathbf{u}_K^1 \\
&= \int_{\mathbf{u}_K^1} q(\mathbf{u}_K^1) \log \frac{p(\mathbf{u}_0 \mid \mathbf{z}, \nu) \prod_{k=0}^{K^1-1} \left| \det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1, \mathbf{Z})}(\mathbf{u}_k^1)}{\partial \mathbf{u}_k^1} \right|^{-1}}{q(\mathbf{u}_0 \mid \mathbf{m}, \mathbf{S}) \prod_{k=0}^{K^1-1} \left| \det \frac{\partial \mathbb{G}^1_{\theta_k(\mathbf{W}^1, \mathbf{Z})}(\mathbf{u}_k^1)}{\partial \mathbf{u}_k^1} \right|^{-1}} \mathrm{d}\mathbf{u}_K^1 \\
&= \int_{\mathbf{u}_0} q(\mathbf{u}_0) \log \frac{p(\mathbf{u}_0 \mid \mathbf{z}, \nu)}{q(\mathbf{u}_0 \mid \mathbf{m}, \mathbf{S})} \mathrm{d}\mathbf{u}_0
\end{aligned}
\tag{42}
$$

which is a similar derivation to that in Maroñas et al. (2021). Note that we could also recognize the LOTUS rule being applied from equality 3 to equality 7 directly in Eq. 41, by previously integrating out $\overline{\mathbf{f}}_K$ and without the $\delta(\cdot)$ cancellations. In other words, we can see the full KL over $\overline{\mathbf{u}}_K$ as a direct reparameterization applied to $\mathbf{u}_0$.

We next derive the ELL:

$$
\begin{aligned}
\text{ELL} &= \int_{\overline{\mathbf{W}}} \int_{\overline{\mathbf{f}}_K} \int_{\overline{\mathbf{u}}_K} q(\overline{\mathbf{f}}_K, \overline{\mathbf{u}}_K \mid \overline{\mathbf{W}}) q(\overline{\mathbf{W}}) \log p(\mathbf{y} \mid \overline{\mathbf{f}}_K) \mathrm{d}\overline{\mathbf{f}}_K \mathrm{d}\overline{\mathbf{u}}_K \mathrm{d}\overline{\mathbf{W}} \\
&= \int_{\overline{\mathbf{W}}} \int_{\overline{\mathbf{f}}_K} q(\overline{\mathbf{f}}_K \mid \overline{\mathbf{W}}) q(\overline{\mathbf{W}}) \log p(\mathbf{y} \mid \overline{\mathbf{f}}_K) \mathrm{d}\overline{\mathbf{f}}_K \mathrm{d}\overline{\mathbf{W}} \\
&= \int_{\overline{\mathbf{W}}} \int_{\mathbf{f}_0} q(\mathbf{f}_0) q(\overline{\mathbf{W}}) \log p(\mathbf{y} \mid \overline{\mathbb{G}}_{\boldsymbol{\theta}(\overline{\mathbf{W}}, \mathbf{X})}(\mathbf{f}_0)) \mathrm{d}\mathbf{f}_0 \mathrm{d}\overline{\mathbf{W}} \\
&= \int_{\overline{\mathbf{W}}} \int_{\mathbf{f}_0} q(\mathbf{f}_0) q(\overline{\mathbf{W}}) \log \prod_{n=1}^{N} \prod_{c=1}^{C} \pi_c(\overline{\mathbb{G}}_{\theta_K(\overline{\mathbf{W}}, \mathbf{x}^n)}(f_{0,n}))^{\mathbb{I}(y^n=c)} \mathrm{d}\mathbf{f}_0 \mathrm{d}\overline{\mathbf{W}} \\
&= \sum_{n=1}^{N} \sum_{c=1}^{C} \mathbb{I}(y^n = c) \int_{\overline{\mathbf{W}}} q(\overline{\mathbf{W}}) \int_{f_{0,n}} q(f_{0,n}) \log \pi_c(\overline{\mathbb{G}}_{\theta_K(\overline{\mathbf{W}}, \mathbf{x}^n)}(f_{0,n})) \mathrm{d}f_{0,n} \mathrm{d}\overline{\mathbf{W}} \\
&\approx \sum_{n=1}^{N} \sum_{c=1}^{C} \mathbb{I}(y^n = c) \frac{1}{S} \sum_{s=1}^{S} \int_{f_{0,n}} q(f_{0,n}) \log \pi_c(\overline{\mathbb{G}}_{\theta_K(\overline{\mathbf{W}}^s, \mathbf{x}^n)}(f_{0,n})) \mathrm{d}f_{0,n}; \overline{\mathbf{W}}_s \sim q(\overline{\mathbf{W}})
\end{aligned}
\tag{43}
$$

where we first integrate out $\overline{\mathbf{u}}_K$ yielding the derived conditional marginal $q(\overline{\mathbf{f}}_K \mid \overline{\mathbf{W}})$ and then apply the LOTUS rule to expectation w.r.t. $q(\overline{\mathbf{f}}_K \mid \overline{\mathbf{W}})$. The remaining steps are similar to SVGP when pluging the specific Categorical Likelihood used in this work.

Using both derivations we recover the ELBO in the main paper (Eq. 9):

$$
\begin{aligned}
\text{ELBO} = &\sum_{n=1}^{N} \sum_{c=1}^{C} \mathbb{I}(y^n = c) \int_{\overline{\mathbf{W}}} q(\overline{\mathbf{W}}) \int_{f_{0,n}} q(f_{0,n}) \log \pi_c(\overline{\mathbb{G}}_{\theta_K(\overline{\mathbf{W}}, \mathbf{x}^n)}(f_{0,n})) \mathrm{d}f_{0,n} \mathrm{d}\overline{\mathbf{W}} \\
&- \text{KL}[q(\mathbf{u}_0) || p(\mathbf{u}_0)] - \text{KL}[q(\overline{\mathbf{W}}) || p(\overline{\mathbf{W}})]
\end{aligned}
\tag{44}
$$

A.2.4. COMPUTATIONAL ADVANTAGES

We highlight differences between our proposed model and SVGPs. First, expectations w.r.t. $q(\mathbf{f}_0)$ in Eq. 44 can be computed with 1-d quadrature. By contrast, the SVGP method cannot use quadrature methods and require Monte Carlo. This makes our algorithm computationally advantageous. On the other side expectations w.r.t. the NN's parameters can be computed using batched matrix multiplications and in practice we use Monte Carlo Dropout (Gal & Ghahramani, 2016) with one Monte Carlo sample for training, making this computation very efficient. Moreover, the number of GP operations is constant with C. To get $q(\mathbf{f}^c)$ in SVGPs one needs a cubic operation to invert $K_\nu^c(\mathbf{Z}, \mathbf{Z})$ and M^2 operation to compute the variational parameters per class and datapoint, giving a complexity of $\mathcal{O}(CM^3 + CNM^2)$. This can be alleviated by sharing K_ν and $\mathbf{Z}$ across GPs, resulting in $\mathcal{O}(M^3 + CNM^2)$, at the cost of limiting expressiveness, as shown in the experiment section. ETGP cost is always $\mathcal{O}(M^3 + NM^2)$ (without considering the NN's computations, which for the architecture presented is often much faster and can be done in parallel to GP operations).

A.3. Proof of Proposition 1

In this section we prove proposition 1, which we restate for clarity.

Proposition 1. The joint conditional distribution of C non-stationary GPs obtained via a linear flow is given by:

$$p(\overline{\mathbf{f}}_K \mid \overline{\mathbf{W}}) = \mathcal{N}\left(\mathbf{f}_K^1 \mid \mathbf{b}^1, \mathbf{A}_1 K_\nu(\mathbf{X},\mathbf{X})\mathbf{A}_1^{\mathrm{T}}\right) \prod_{c=2}^{C} \delta\left(\mathbf{f}_K^c - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c,\mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1)\right), \tag{45}$$

with $\mathbf{b}^1 = (b^1(\mathbf{x}^1), \dots, b^1(\mathbf{x}^N))^{\mathrm{T}}$ and $\mathbf{A}_1 \in \mathbb{R}^{N\times N}$ a diagonal matrix with entries $\mathbf{a}^1 = (a^1(\mathbf{x}^1), \dots, a^1(\mathbf{x}^N))^{\mathrm{T}}$. Each marginal $p(\mathbf{f}_K^c)$ is Gaussian with mean and covariance given by $\mathbf{b}^c$ and $\mathbf{A}_c K_\nu(\mathbf{X},\mathbf{X})\mathbf{A}_c^{\mathrm{T}}$, respectively. The covariance matrix between the pivot $\mathbf{f}_K^1$ and $\mathbf{f}_K^c$, for $c \neq 1$, is given by: $\mathbf{a}^1(\mathbf{a}^c)^{\mathrm{T}} \odot K_\nu(\mathbf{X},\mathbf{X})$ with $\odot$ denoting Hadamart product.

The proof is divided in two steps. We first derive the marginal distributions $\{p(\mathbf{f}_K^c)\}_{c=1}^C$ and then the covariances. For all the proof we will assume that the *pivot* is $\mathbf{f}_K^1$. First, since a linear flow is used, we can write the flow mapping over a set of samples $\mathbf{X} = (\mathbf{x}^1, \dots, \mathbf{x}^N)$ in matrix form as:

$$\mathbf{f}_K^1 = \mathbf{A}_1\mathbf{f}_0 + \mathbf{b}^1 \tag{46}$$

with $\mathbf{A}_1 \in \mathbb{R}^{N\times N}$ being a diagonal matrix with entries $\mathbf{a}^1 = (a^1(\mathbf{x}^1), \dots, a^1(\mathbf{x}^N))^{\mathrm{T}}$ and $\mathbf{b}^1 = (b^1(\mathbf{x}^1), \dots, b^1(\mathbf{x}^N))^{\mathrm{T}}$. Thus, using the fact that $p(\mathbf{f}_0 \mid \mathbf{X}, \nu) = \mathcal{N}(\mathbf{f}_0 \mid \mathbf{0}, K_\nu(\mathbf{X},\mathbf{X}))$ and the resulting density when applying a linear transformation to a Gaussian density, the marginal distribution over $\mathbf{f}_K^1$ is:

$$\mathcal{N}\left(\mathbf{f}_K^1 \mid \mathbf{b}^1, \mathbf{A}_1 K_\nu(\mathbf{X},\mathbf{X})\mathbf{A}_1^{\mathrm{T}}\right) \tag{47}$$

Note that for non-zero mean GP the mean would be given by $\mathbf{b}^1 + \mathbf{A}_1\mu_\nu(\mathbf{X})$. To derive the marginal distribution for each c we solve the following integral:

$$\begin{aligned} p(\mathbf{f}_K^c) &= \int_{\mathbf{f}_K^1} p(\mathbf{f}_K^1) p(\mathbf{f}_K^c \mid \mathbf{f}_K^1) \mathrm{d}\mathbf{f}_K^1 \\ &= \int_{\mathbf{f}_K^1} \mathcal{N}\left(\mathbf{f}_K^1 \mid \mathbf{b}^1, \mathbf{A}_1 K_\nu(\mathbf{X},\mathbf{X})\mathbf{A}_1^{\mathrm{T}}\right) \delta\left(\mathbf{f}_K^c - \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c,\mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1)\right) \mathrm{d}\mathbf{f}_K^1 \end{aligned} \tag{48}$$

Before solving it note that for any c we have:

$$\begin{aligned} \mathbf{f}_K^c &= \mathbb{G}^c_{\boldsymbol{\theta}(\mathbf{W}^c,\mathbf{X})} \circ \mathbb{H}^1_{\boldsymbol{\theta}(\mathbf{W}^1,\mathbf{X})}(\mathbf{f}_K^1) \\ &= \overbrace{\mathbf{A}_c \underbrace{\mathbf{A}_1^{-1}\left[\mathbf{f}_K^1 - \mathbf{b}^1\right]}_{\mathbb{H}} + \mathbf{b}^c}^{\mathbb{G}} \end{aligned} \tag{49}$$

yielding:

$$\begin{aligned} p(\mathbf{f}_K^c) &= \int_{\mathbf{f}_K^1} \mathcal{N}\left(\mathbf{f}_K^1 \mid \mathbf{b}^1, \mathbf{A}_1 K_\nu(\mathbf{X},\mathbf{X})\mathbf{A}_1^{\mathrm{T}}\right) \delta(\mathbf{f}_K^c - \mathbf{A}_c\mathbf{A}_1^{-1}\left[\mathbf{f}_K^1 - \mathbf{b}^1\right] - \mathbf{b}^c) \mathrm{d}\mathbf{f}_K^1 \\ &= \int_{\mathbf{f}_K^1} \mathcal{N}\left(\mathbf{f}_K^1 \mid \mathbf{b}^1, \mathbf{A}_1 K_\nu(\mathbf{X},\mathbf{X})\mathbf{A}_1^{\mathrm{T}}\right) \delta(\mathbf{f}_K^c - \mathbf{A}_c\mathbf{A}_1^{-1}\mathbf{f}_K^1 + \mathbf{A}_c\mathbf{A}_1^{-1}\mathbf{b}^1 - \mathbf{b}^c) \mathrm{d}\mathbf{f}_K^1 \end{aligned} \tag{50}$$

We then rewrite this last expression to highlight the integral to be solved:

$$\int_{\mathbf{f}_K^1} \mathcal{N}\left(\mathbf{f}_K^1 \mid \mathbf{m}, \mathbf{S}\right) \delta(\mathbf{f}_K^c - \mathbf{Q}\mathbf{f}_K^1 - \mathbf{r}) \mathrm{d}\mathbf{f}_K^1 \tag{51}$$

with:

$$\begin{aligned} \mathbf{m} &= \mathbf{b}^1 \\ \mathbf{S} &= \mathbf{A}_1 K_\nu(\mathbf{X},\mathbf{X})\mathbf{A}_1^{\mathrm{T}} \\ \mathbf{Q} &= \mathbf{A}_c\mathbf{A}_1^{-1} \\ \mathbf{r} &= -\mathbf{A}_c\mathbf{A}_1^{-1}\mathbf{b}^1 + \mathbf{b}^c \end{aligned} \tag{52}$$

The solution of this integral is obtained by the following procedure. First note that if $\mathbf{Q} = \mathbf{I}$ we recognize a convolution between a Gaussian and a Dirac delta function, easily solved by applying the selection property of Dirac delta functions:

$$\begin{aligned}\mathcal{N}\left(\mathbf{f}_K^c \mid \mathbf{m}, \mathbf{S}\right) \circledast \delta(\mathbf{f}_K^c - \mathbf{r}) &\coloneqq \int_{-\infty}^{\infty} \mathcal{N}\left(\mathbf{f}_K^1 \mid \mathbf{m}, \mathbf{S}\right) \delta(\mathbf{f}_K^c - \mathbf{f}_K^1 - \mathbf{r}) \mathrm{d}\mathbf{f}_K^1 \\ = \mathcal{N}\left(\mathbf{f}_K^c - \mathbf{r} \mid \mathbf{m}, \mathbf{S}\right) &= \mathcal{N}\left(\mathbf{f}_K^c \mid \mathbf{m} + \mathbf{r}, \mathbf{S}\right)\end{aligned} \tag{53}$$

where the last step holds by writing the Gaussian density and checking $\mathbf{f}_K^c - \mathbf{r} - \mathbf{m} = \mathbf{f}_K^c - (\mathbf{m} + \mathbf{r})$. For $\mathbf{Q} \neq \mathbf{I}$, we perform an integration by substitution [6], since there is no way we can write the integral as a convolution between two functions. More precisely let $u = \mathbf{Q}\mathbf{f}_K^1 + \mathbf{r}$. We have $\mathbf{f}_K^1 = \mathbf{Q}^{-1}(u - \mathbf{r})$ and $|\det {}^{\mathrm{d}u}/_{\mathrm{d}\mathbf{f}_K^1}| = |\det \mathbf{Q}|$ which implies the substitution $\mathrm{d}\mathbf{f}_K^1 = |^1/_{\det \mathbf{Q}}| \mathrm{d}u$. Putting all together we have:

$$\begin{aligned} p(\mathbf{f}_K^c) &= \int_{\mathbf{f}_K^1} \mathcal{N}\left(\mathbf{f}_K^1 \mid \mathbf{m}, \mathbf{S}\right) \delta(\mathbf{f}_K^c - \mathbf{Q}\mathbf{f}_K^1 - \mathbf{r}) \mathrm{d}\mathbf{f}_K^1 \\ &= \int_{\mathbf{f}_K^1} \mathcal{N}\left(\mathbf{Q}^{-1}(u - \mathbf{r}) \mid \mathbf{m}, \mathbf{S}\right) \delta(\mathbf{f}_K^c - u) \left|\frac{1}{\det \mathbf{Q}}\right| \mathrm{d}u \\ &= \frac{1}{|\det \mathbf{Q}|} \mathcal{N}\left(\mathbf{Q}^{-1}(\mathbf{f}_K^c - \mathbf{r}) \mid \mathbf{m}, \mathbf{S}\right) \end{aligned} \tag{54}$$

Beyond the substitution the integral is solved by applying the selection property of the delta function. After applying some standard algebra to the Gaussian distribution (see App. A.5) we have:

$$\mathcal{N}\left(\mathbf{Q}^{-1}(\mathbf{f}_K^c - \mathbf{r}) \mid \mathbf{m}, \mathbf{S}\right) = |\det \mathbf{Q}| \, \mathcal{N}\left(\mathbf{f}_K^c \mid \mathbf{Q}\mathbf{m} + \mathbf{r}, \mathbf{Q}\mathbf{S}\mathbf{Q}^{\mathrm{T}}\right) \tag{55}$$

Giving the final result:

$$p(\mathbf{f}_K^c) = \mathcal{N}\left(\mathbf{f}_K^c \mid \mathbf{Q}\mathbf{m} + \mathbf{r}, \mathbf{Q}\mathbf{S}\mathbf{Q}^{\mathrm{T}}\right) \tag{56}$$

since $|\det \mathbf{Q}|$ cancels with $^1/_{|\det \mathbf{Q}|}$. Note this result matches the one obtained with the convolution for $\mathbf{Q} = \mathbf{I}$. If we now substitute the shortcuts for $\mathbf{Q}, \mathbf{S}, \mathbf{m}, \mathbf{r}$ we have for the covariance:

$$\begin{aligned} \mathbf{Q}\mathbf{S}\mathbf{Q}^{\mathrm{T}} &= \mathbf{A}_c \mathbf{A}_1^{-1} \mathbf{A}_1 K_\nu(\mathbf{X}, \mathbf{X}) \mathbf{A}_1^{\mathrm{T}} \left[\mathbf{A}_c \mathbf{A}_1^{-1}\right]^{\mathrm{T}} \\ &= \mathbf{A}_c \mathbf{A}_1^{-1} \mathbf{A}_1 K_\nu(\mathbf{X}, \mathbf{X}) \mathbf{A}_1 \mathbf{A}_1^{-1} \mathbf{A}_c \\ &= \mathbf{A}_c K_\nu(\mathbf{X}, \mathbf{X}) \mathbf{A}_c \end{aligned} \tag{57}$$

where we apply some standard matrix identities. In particular the transpose of the product is the product of the transposes in reverse order, and the transpose of a diagonal matrix is equal to the diagonal matrix. For the mean we have:

$$\mathbf{r} + \mathbf{Q}\mathbf{m} = \cancel{-\mathbf{A}_c \mathbf{A}_1^{-1} \mathbf{b}^1} + \mathbf{b}^c \cancel{+\mathbf{A}_c \mathbf{A}_1^{-1} \mathbf{b}^1} = \mathbf{b}^c \tag{58}$$

finishing the first part of the proof, which we re-emphasize:

The marginal distribution for any $\mathbf{f}_K^c$ has density given by: $\mathcal{N}\left(\mathbf{f}_K^c \mid \mathbf{b}^c, \mathbf{A}_c K_\nu(\mathbf{X}, \mathbf{X}) \mathbf{A}_c^T\right)$

Importantly, note that if non-zero GPs are used, this result is also matched with the particularity that the mean is given by $\mathbf{b}^c + \mathbf{A}_c \mu_\nu(\mathbf{X})$. This is seen by replacing $\mathbf{A}_c \mathbf{A}_1^{-1} \mathbf{b}^1$ for $\mathbf{A}_c \mathbf{A}_1^{-1} [\mathbf{b}^1 + \mathbf{A}_1 \mu_\nu(\mathbf{X})]$ in Eq. 58.

The second part of the proof characterizes some of the linear dependencies (covariance) in the joint distribution $p(\overline{\mathbf{f}}_K)$. We note that in this paper we have not figured out the form of the joint distribution if the *pivot* is integrated out, i.e. how the pivot couples the rest of latent functions. In other words, the distribution $p(\mathbf{f}_K^2, \ldots, \mathbf{f}_K^C) = \int_{\mathbf{f}_K^1} p(\mathbf{f}_K^2, \ldots, \mathbf{f}_K^C \mid \mathbf{f}_K^1) p(\mathbf{f}_K^1) \mathrm{d}\mathbf{f}_K^1$ is unknown, which limits the full characterization of the covariances in the joint distribution.

For this reason, in this proposition we just characterize the covariances between any $\mathbf{f}_K^c$ and the *pivot* $\mathbf{f}_K^1$. For this we use the expression:

$$\mathbb{COV}[\mathbf{f}_K^1, \mathbf{f}_K^c] = \mathbb{E}[\mathbf{f}_K^1 (\mathbf{f}_K^c)^{\mathrm{T}}] - \mathbb{E}[\mathbf{f}_K^1] \mathbb{E}[\mathbf{f}_K^c]^{\mathrm{T}} \tag{59}$$

[6] After this derivation we found a simpler way to obtain this solution which is given, for completeness, in App. A.4.

The expected values can be directly obtained from the marginal distributions obtained in the first part of the proof. In particular:

$$\begin{aligned}\mathbb{E}\left[\mathbf{f}_K^1\right] &= \mathbf{b}^1 \\ \mathbb{E}\left[\mathbf{f}_K^c\right] &= \mathbf{b}^c\end{aligned} \tag{60}$$

To derive the covariance, it is easier to do it by just looking at its entries at two single points locations $\mathbf{x}^n$ and $\mathbf{x}^{n'}$ and then generalizing the result. Following the main paper notation we have $f_{0,n} \coloneqq f_0(\mathbf{x}^n)$, $b^c(\mathbf{x}^n) \coloneqq b_n^c$ and $a^c(\mathbf{x}^n) \coloneqq a_n^c$.

A.3.1. Covariance Between $f_{K,n}^1$ and $f_{K,n}^c$ at a Single Location n

For this we compute:

$$\begin{aligned}&\mathbb{E}\left[f_{K,n}^1 f_{K,n}^c\right] = \\ &\int_{f_{K,n}^1}\int_{f_{K,n}^c} f_{K,n}^1 f_{K,n}^c \mathcal{N}\left(f_{K,n}^1 \mid b_n^1, a_n^1 K_\nu(\mathbf{x}^n,\mathbf{x}^n)a_n^1\right)\delta\left(f_{K,n}^c - \frac{a_n^c}{a_n^1}\left[f_{K,n}^1 - b_n^1\right] - b_n^c\right)\mathrm{d}f_{K,n}^1\mathrm{d}f_{K,n}^c \\ &= \int_{f_{K,n}^1}\left[\frac{a_n^c}{a_n^1}f_{K,n}^1 f_{K,n}^1 - \frac{a_n^c}{a_n^1}b_n^1 f_{K,n}^1 + b_n^c f_{K,n}^1\right]\mathcal{N}\left(f_{K,n}^1 \mid b_n^1, a_n^1 K_\nu(\mathbf{x}^n,\mathbf{x}^n)a_n^1\right)\mathrm{d}f_{K,n}^1 \\ &= \frac{a_n^c}{a_n^1}\left[a_n^1 K_\nu(\mathbf{x}^n,\mathbf{x}^n)a_n^1 + \left(b_n^1\right)^2\right] - \frac{a_n^c}{a_n^1}\left(b_n^1\right)^2 + b_n^c b_n^1 \\ &= a_n^c K_\nu(\mathbf{x}^n,\mathbf{x}^n)a_n^1 + b_n^c b_n^1\end{aligned} \tag{61}$$

yielding,

$$\begin{aligned}\mathbb{COV}[f_{K,n}^1, f_{K,n}^c] &= \mathbb{E}\left[f_{K,n}^1 f_{K,n}^c\right] - \mathbb{E}\left[f_{K,n}^1\right]\mathbb{E}\left[f_{K,n}^c\right] \\ &= a_n^c K_\nu(\mathbf{x}^n,\mathbf{x}^n)a_n^1 + b_n^c b_n^1 - b_n^c b_n^1 \\ &= a_n^c K_\nu(\mathbf{x}^n,\mathbf{x}^n)a_n^1\end{aligned} \tag{62}$$

So the covariance between two processes at all locations N will be given by the diagonal of $K_\nu(\mathbf{X},\mathbf{X})$ element-wise multiplied by the diagonal of $\mathbf{a}^1(\mathbf{a}^c)^{\mathrm{T}}$.

A.3.2. Covariance Between $f_{K,n}^1$ and $f_{K,n'}^c$ at Different Locations n, n'

We again emphasize the main paper notation for easier presentation and re-highlight that for this particular section $\mathbf{f}_K^1 = (f_{K,n}^1, f_{K,n'}^1)^{\mathrm{T}}$ and similar for any parameter short cut e.g. $\mathbf{b}^1$. For this we compute:

$$\begin{aligned}&\mathbb{E}\left[f_{K,n}^1 f_{K,n'}^c\right] = \\ &\int_{f_{K,n}^1}\int_{f_{K,n'}^1}\int_{f_{K,n'}^c} f_{K,n}^1 f_{K,n'}^c \mathcal{N}\left(\mathbf{f}_K^1 \mid \mathbf{b}^1, \mathbf{A}_1 K_\nu(\mathbf{X},\mathbf{X})\mathbf{A}_1^{\mathrm{T}}\right)\delta\left(f_{K,n'}^c - \frac{a_{n'}^c}{a_{n'}^1}\left[f_{K,n'}^1 - b_{n'}^1\right] - b_{n'}^c\right)\mathrm{d}f_{K,n}^1\mathrm{d}f_{K,n'}^1\mathrm{d}f_{K,n'}^c \\ &= \int_{f_{K,n}^1}\int_{f_{K,n'}^1}\left[\frac{a_{n'}^c}{a_{n'}^1}f_{K,n}^1 f_{K,n'}^1 - \frac{a_{n'}^c}{a_{n'}^1}b_{n'}^1 f_{K,n}^1 + b_{n'}^c f_{K,n}^1\right]\mathcal{N}\left(\mathbf{f}_K^1 \mid \mathbf{b}^1, \mathbf{A}_1 K_\nu(\mathbf{X},\mathbf{X})\mathbf{A}_1^{\mathrm{T}}\right)\mathrm{d}f_{K,n}^1\mathrm{d}f_{K,n'}^1 \\ &= \frac{a_{n'}^c}{a_{n'}^1}\left[a_n^1 K_\nu(\mathbf{x}^n,\mathbf{x}^{n'})a_{n'}^1 + b_{n'}^1 b_n^1\right] - \frac{a_{n'}^c}{a_{n'}^1}b_{n'}^1 b_n^1 + b_{n'}^c b_n^1 \\ &= a_{n'}^c a_n^1 K_\nu(\mathbf{x}^n,\mathbf{x}^{n'}) + b_{n'}^c b_n^1\end{aligned} \tag{63}$$

yielding,

$$\begin{aligned}\mathbb{COV}[f_{K,n}^1, f_{K,n'}^c] &= \mathbb{E}\left[f_{K,n}^1 f_{K,n'}^c\right] - \mathbb{E}\left[f_{K,n}^1\right]\mathbb{E}\left[f_{K,n'}^c\right] \\ &= a_{n'}^c a_n^1 K_\nu(\mathbf{x}^n,\mathbf{x}^{n'}) + b_{n'}^c b_n^1 - b_n^1 b_{n'}^c \\ &= a_{n'}^c a_n^1 K_\nu(\mathbf{x}^n,\mathbf{x}^{n'})\end{aligned} \tag{64}$$

Note that from the above integral is easy to see that $\mathbb{COV}[f_{K,n'}^1, f_{K,n}^c] = a_n^c a_{n'}^1 K_\nu(\mathbf{x}^{n'},\mathbf{x}^n)$.

A.3.3. COVARIANCE BETWEEN $\mathbf{f}_K^1$ AND $\mathbf{f}_K^c$

We have derived the covariance between the *pivot* and any other process at the same location $\mathbf{x}^n$ and between two different locations $\mathbf{x}^n, \mathbf{x}^{n'}$. Note that for N arbitrary locations $\mathbf{X}$, the covariance between any pair of elements can be obtained by any of the two derivations shown above.

In summary, at a given location $\mathbf{x}^n$ the covariance between two processes is $a^c(\mathbf{x}^n)a^1(\mathbf{x}^n)K_\nu(\mathbf{x}^n, \mathbf{x}^n)$ and at two different locations $\mathbf{x}^n$ and $\mathbf{x}^{n'}$ the covariance between two processes is $a^c(\mathbf{x}^{n'})a^1(\mathbf{x}^n)K_\nu(\mathbf{x}^n, \mathbf{x}^{n'})$.

Thus, the covariance abetween the processes at N locations $\mathbf{X}$ is given by $\mathbf{a}^1(\mathbf{a}^c)^\mathrm{T} \odot K_\nu(\mathbf{X}, \mathbf{X})$.

A.4. Alternative Derivation of Eq. 56

We found a simpler way of obtaining Eq. 56 by approximating the Dirac measure with a Gaussian and taking the limit of the variance to 0. With this, we can apply standard Gaussian integration and yield the desired result.

First:

$$\delta(\mathbf{f}_K^c - \mathbf{Q}\mathbf{f}_K^1 - \mathbf{r}) = \lim_{\lambda \to 0} \mathcal{N}\left(\mathbf{f}_K^c \mid \mathbf{Q}\mathbf{f}_K^1 + \mathbf{r}, \lambda\mathbf{I}\right) \tag{65}$$

and so the integral is solved by:

$$\begin{aligned} p(\mathbf{f}_K^c) &= \lim_{\lambda \to 0} \int_{\mathbf{f}_K^1} \mathcal{N}\left(\mathbf{f}_K^1 \mid \mathbf{m}, \mathbf{S}\right) \mathcal{N}\left(\mathbf{f}_K^c \mid \mathbf{Q}\mathbf{f}_K^1 + \mathbf{r}, \lambda\mathbf{I}\right) \mathrm{d}\mathbf{f}_K^1 \\ &= \lim_{\lambda \to 0} \mathcal{N}\left(\mathbf{f}_K^c \mid \mathbf{Q}\mathbf{m} + \mathbf{r}, \lambda\mathbf{I} + \mathbf{Q}\mathbf{S}\mathbf{Q}^\mathrm{T}\right) \\ &= \mathcal{N}\left(\mathbf{f}_K^c \mid \mathbf{Q}\mathbf{m} + \mathbf{r}, \mathbf{Q}\mathbf{S}\mathbf{Q}^\mathrm{T}\right) \end{aligned} \tag{66}$$

A.5. Algebra Manipulation of the Gaussian Distribution

We are interested in showing:

$$\mathcal{N}\left(\mathbf{Q}^{-1}(\mathbf{f}_K - \mathbf{r}) \mid \mathbf{m}, \mathbf{S}\right) = \det \mathbf{Q}\, \mathcal{N}\left(\mathbf{f}_K \mid \mathbf{Q}\mathbf{m} + \mathbf{r}, \mathbf{Q}\mathbf{S}\mathbf{Q}^\mathrm{T}\right) \tag{67}$$

We provide the steps to be performed since we haven't found the steps available in the references searched. We can show this equivalence either by the technique of completing the square or by making simple manipulations to the exponent in the Gaussian distribution. We assume these Gaussian distributions have dimensionality n. Our manipulations use standard matrix operations that can be found in the matrix cookbook (Petersen & Pedersen, 2012).

A.5.1. MANIPULATION OF THE EXPONENT

We have:

$$\begin{aligned} &\mathcal{N}\left(\mathbf{Q}^{-1}(\mathbf{f}_K - \mathbf{r}) \mid \mathbf{m}, \mathbf{S}\right) \\ &= \frac{1}{(2\pi)^{n/2}(\det \mathbf{S})^{1/2}} \exp\left\{(\mathbf{Q}^{-1}(\mathbf{f}_K - \mathbf{r}) - \mathbf{m})^\mathrm{T}\mathbf{S}^{-1}(\mathbf{Q}^{-1}(\mathbf{f}_K - \mathbf{r}) - \mathbf{m})\right\} \\ &= \frac{1}{(2\pi)^{n/2}(\det \mathbf{S})^{1/2}} \exp\left\{((\mathbf{Q}^{-1}\mathbf{f}_K)^\mathrm{T} - (\mathbf{Q}^{-1}\mathbf{r})^\mathrm{T} - \mathbf{m}^\mathrm{T})\mathbf{Q}^\mathrm{T}(\mathbf{Q}^\mathrm{T})^{-1}\mathbf{S}^{-1}\mathbf{Q}^{-1}\mathbf{Q}(\mathbf{Q}^{-1}\mathbf{f}_K - \mathbf{Q}^{-1}\mathbf{r} - \mathbf{m})\right\} \\ &= \frac{1}{(2\pi)^{n/2}(\det \mathbf{S})^{1/2}} \exp\left\{((\mathbf{Q}^{-1}\mathbf{f}_K)^\mathrm{T}\mathbf{Q}^\mathrm{T} - (\mathbf{Q}^{-1}\mathbf{r})^\mathrm{T}\mathbf{Q}^\mathrm{T} - \mathbf{m}^\mathrm{T}\mathbf{Q}^\mathrm{T})(\mathbf{Q}^{-1})^\mathrm{T}\mathbf{S}^{-1}\mathbf{Q}^{-1}(\mathbf{f}_K - \mathbf{r} - \mathbf{Q}\mathbf{m})\right\} \\ &= \frac{1}{(2\pi)^{n/2}(\det \mathbf{S})^{1/2}} \exp\left\{(\mathbf{f}_K{}^\mathrm{T}(\mathbf{Q}^\mathrm{T})^{-1}\mathbf{Q}^\mathrm{T} - \mathbf{r}^\mathrm{T}(\mathbf{Q}^\mathrm{T})^{-1}\mathbf{Q}^\mathrm{T} - (\mathbf{Q}\mathbf{m})^\mathrm{T})(\mathbf{Q}^{-1})^\mathrm{T}\mathbf{S}^{-1}\mathbf{Q}^{-1}(\mathbf{f}_K - \mathbf{r} - \mathbf{Q}\mathbf{m})\right\} \\ &= \frac{1}{(2\pi)^{n/2}(\det \mathbf{S})^{1/2}} \exp\left\{(\mathbf{f}_K{}^\mathrm{T} - \mathbf{r}^\mathrm{T} - (\mathbf{Q}\mathbf{m})^\mathrm{T})(\mathbf{Q}\mathbf{S}\mathbf{Q}^\mathrm{T})^{-1}(\mathbf{f}_K - \mathbf{r} - \mathbf{Q}\mathbf{m})\right\} \\ &= \frac{1}{(2\pi)^{n/2}(\det \mathbf{S})^{1/2}} \exp\left\{(\mathbf{f}_K - \mathbf{r} - \mathbf{Q}\mathbf{m})^\mathrm{T}(\mathbf{Q}\mathbf{S}\mathbf{Q}^\mathrm{T})^{-1}(\mathbf{f}_K - \mathbf{r} - \mathbf{Q}\mathbf{m})\right\} \end{aligned} \tag{68}$$

This gives an unnormalized Gaussian distribution with mean $\mathbf{Q}\mathbf{m} + \mathbf{r}$ and covariance $\mathbf{Q}\mathbf{S}\mathbf{Q}^\mathrm{T}$

A.5.2. COMPLETING THE SQUARE METHOD

We can obtain a similar result by the technique of completing the square. Since we know that a scale and shift on a function argument does not change the function shape, i.e. scaling and shifting a Gaussian will give a Gaussian curve, we can use the technique of completing the square to recognize the mean and covariance matrix (Bishop, 2007).

From:

$$\begin{aligned}&\mathcal{N}\left(\mathbf{Q}^{-1}(\mathbf{f}_K-\mathbf{r})\mid\mathbf{m},\mathbf{S}\right)\\&=\frac{1}{(2\pi)^{n/2}(\det\mathbf{S})^{1/2}}\exp\left\{(\mathbf{Q}^{-1}(\mathbf{f}_K-\mathbf{r})-\mathbf{m})^{\mathrm{T}}\mathbf{S}^{-1}(\mathbf{Q}^{-1}(\mathbf{f}_K-\mathbf{r})-\mathbf{m})\right\}\end{aligned}\tag{69}$$

We expand the quadratic form:

$$\begin{aligned}&(\mathbf{Q}^{-1}(\mathbf{f}_K-\mathbf{r})-\mathbf{m})^{\mathrm{T}}\mathbf{S}^{-1}(\mathbf{Q}^{-1}(\mathbf{f}_K-\mathbf{r})-\mathbf{m})\\&=(\mathbf{Q}^{-1}(\mathbf{f}_K-\mathbf{r}))^{\mathrm{T}}\mathbf{S}^{-1}(\mathbf{Q}^{-1}(\mathbf{f}_K-\mathbf{r}))-2(\mathbf{Q}^{-1}(\mathbf{f}_K-\mathbf{r}))^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{m}+\mathbf{m}^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{m}\\&=(\mathbf{Q}^{-1}\mathbf{f}_K)^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{Q}^{-1}\mathbf{f}_K-(\mathbf{Q}^{-1}\mathbf{f}_K)^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{Q}^{-1}\mathbf{r}-(\mathbf{Q}^{-1}\mathbf{r})^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{Q}^{-1}\mathbf{f}_K\\&+(\mathbf{Q}^{-1}\mathbf{r})^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{Q}^{-1}\mathbf{r}-2(\mathbf{Q}^{-1}\mathbf{f}_K)^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{m}+2(\mathbf{Q}^{-1}\mathbf{r})^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{m}+\mathbf{m}^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{m}\end{aligned}\tag{70}$$

and first recognize the terms depending quadratically on $\mathbf{f}_K$:

$$\begin{aligned}(\mathbf{Q}^{-1}\mathbf{f}_K)^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{Q}^{-1}\mathbf{f}_K&=\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}^{-1})^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{Q}^{-1}\mathbf{f}_K\\&=\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}^{\mathrm{T}})^{-1}\mathbf{S}^{-1}\mathbf{Q}^{-1}\mathbf{f}_K\\&=\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}\mathbf{S}\mathbf{Q}^{\mathrm{T}})^{-1}\mathbf{f}_K\end{aligned}\tag{71}$$

Recognizing the covariance to be $\mathbf{Q}\mathbf{S}\mathbf{Q}^{\mathrm{T}}$. Now looking at the terms that depend linearly on $\mathbf{f}_K$ we have:

$$\begin{aligned}&-(\mathbf{Q}^{-1}\mathbf{f}_K)^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{Q}^{-1}\mathbf{r}-(\mathbf{Q}^{-1}\mathbf{r})^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{Q}^{-1}\mathbf{f}_K-2(\mathbf{Q}^{-1}\mathbf{f}_K)^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{m}\\&=-\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}^{-1})^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{Q}^{-1}\mathbf{r}-\mathbf{r}^{\mathrm{T}}(\mathbf{Q}^{-1})^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{Q}^{-1}\mathbf{f}_K-2\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}^{-1})^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{m}\\&=-\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}\mathbf{S}\mathbf{Q}^{\mathrm{T}})^{-1}\mathbf{r}-\mathbf{r}^{\mathrm{T}}(\mathbf{Q}\mathbf{S}\mathbf{Q}^{\mathrm{T}})^{-1}\mathbf{f}_K-2\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}^{-1})^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{m}\end{aligned}\tag{72}$$

Looking closely at $\mathbf{r}^{\mathrm{T}}(\mathbf{Q}\mathbf{S}\mathbf{Q}^{\mathrm{T}})^{-1}\mathbf{f}_K$ we have:

$$\begin{aligned}\mathbf{r}^{\mathrm{T}}(\mathbf{Q}\mathbf{S}\mathbf{Q}^{\mathrm{T}})^{-1}\mathbf{f}_K&=\mathbf{f}_K{}^{\mathrm{T}}((\mathbf{Q}\mathbf{S}\mathbf{Q}^{\mathrm{T}})^{-1})^{\mathrm{T}}\mathbf{r}\\&=\mathbf{f}_K{}^{\mathrm{T}}((\mathbf{Q}\mathbf{S}\mathbf{Q}^{\mathrm{T}})^{\mathrm{T}})^{-1}\mathbf{r}\\&=\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}\mathbf{S}^{\mathrm{T}}\mathbf{Q}^{\mathrm{T}})^{-1}\mathbf{r}\\&=\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}\mathbf{S}\mathbf{Q}^{\mathrm{T}})^{-1}\mathbf{r}\end{aligned}\tag{73}$$

since $\mathbf{S}$ is symmetric i.e. $\mathbf{S}=\mathbf{S}^{\mathrm{T}}$. This yields a final linear term given by:

$$-\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}\mathbf{S}\mathbf{Q}^{\mathrm{T}})^{-1}\mathbf{r}-\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}\mathbf{S}\mathbf{Q}^{\mathrm{T}})^{-1}\mathbf{r}-2\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}^{-1})^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{m}\tag{74}$$

By rewriting $-2\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}^{-1})^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{m}$ as:

$$\begin{aligned}&-2\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}^{-1})^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{m}=-2\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}^{-1})^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{Q}^{-1}\mathbf{Q}\mathbf{m}\\&=-2\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}\mathbf{S}\mathbf{Q}^{\mathrm{T}})^{-1}\mathbf{Q}\mathbf{m}\end{aligned}\tag{75}$$

We obtain the final desired linear term:

$$\begin{aligned}&-\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}\mathbf{S}\mathbf{Q}^{\mathrm{T}})^{-1}\mathbf{r}-\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}\mathbf{S}\mathbf{Q}^{\mathrm{T}})^{-1}\mathbf{r}-2\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}^{-1})^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{m}=\\&-2\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}\mathbf{S}\mathbf{Q}^{\mathrm{T}})^{-1}\mathbf{r}-2\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}\mathbf{S}\mathbf{Q}^{\mathrm{T}})^{-1}\mathbf{Q}\mathbf{m}\\&=-2\mathbf{f}_K{}^{\mathrm{T}}(\mathbf{Q}\mathbf{S}\mathbf{Q}^{\mathrm{T}})^{-1}[\mathbf{Q}\mathbf{m}+\mathbf{r}]\end{aligned}\tag{76}$$

Recognizing $\mathbf{Q}\mathbf{m}+\mathbf{r}$ as the mean of the distribution.

Finally, we would have to check if there is a way for the remaining constant terms to be re-written as $[\mathbf{Qm}+\mathbf{r}]^{\mathrm{T}}\mathbf{S}^{-1}[\mathbf{Qm}+\mathbf{r}]$ otherwise the probability density would not be correctly normalized. Since we know from the previous section that this yields an unnormalized density we omit this step. Anyway we can skip this step since we know that function argument scaling is a non-volume preserving transformation, something that can be trivially checked by computing the area of a function $x(t) = u(t) - u(t-1)$ and its scaled version $x(2t)$, where $u(t)$ is the unit (or Heaviside) step function.

This can be more formally check if we consider the probability under change of variable. If we have $\mathbf{y} = f(\mathbf{x})$ for some invertible function $f(\cdot)$ with $\mathbf{y} \sim p(\mathbf{y})$, then:

$$p(\mathbf{x}) = p(\mathbf{y} = f(\mathbf{x})) \left| \det \frac{\nabla f(\mathbf{x})}{\mathbf{x}} \right| \tag{77}$$

for a linear transformation $\mathbf{y} = \mathbf{Qx} + \mathbf{r}$, this gives:

$$p(\mathbf{x}) = p(\mathbf{y} = \mathbf{Qx} + \mathbf{r}) \left| \det \mathbf{Q} \right| \tag{78}$$

formally showing that a scale in the argument of a density implies a non-volume preserving transformation and thus without the Jacobian correction it would not be a proper normalized density.

A.5.3. The remaining normalization constant

It is clear that the scaling of the Gaussian argument gives an unnormalized density with mean $\mathbf{Qm} + \mathbf{r}$ and covariance $\mathbf{QSQ}^{\mathrm{T}}$. A proper normalized Gaussian density would have a multiplication constant equal to:

$$\frac{1}{(2\pi)^{n/2}(\det \mathbf{QSQ}^{\mathrm{T}})^{1/2}} \tag{79}$$

but our result has:

$$\frac{1}{(2\pi)^{n/2}(\det \mathbf{S})^{1/2}} \tag{80}$$

Operating the determinant we see:

$$\begin{aligned} (\det \mathbf{QSQ}^{\mathrm{T}})^{1/2} &= (\det \mathbf{Q} \det \mathbf{S} \det \mathbf{Q}^{\mathrm{T}})^{1/2} \\ &= (\det \mathbf{S})^{1/2}(\det \mathbf{Q} \det \mathbf{Q})^{1/2} = (\det \mathbf{S})^{1/2} |\det \mathbf{Q}| \end{aligned} \tag{81}$$

where since the determinant is a scalar value we know that $\sqrt{x^2} = |x|$. This means that we need to scale our density by $1/|\det \mathbf{Q}|$, or, in other words, our density has been unnormalized by multiplying it by $|\det \mathbf{Q}|$. With this, we conclude:

$$\mathcal{N}\left(\mathbf{Q}^{-1}(\mathbf{f}_K - \mathbf{r}) \mid \mathbf{m}, \mathbf{S}\right) = |\det \mathbf{Q}| \, \mathcal{N}\left(\mathbf{f}_K \mid \mathbf{Qm} + \mathbf{r}, \mathbf{QSQ}^{\mathrm{T}}\right) \tag{82}$$

B. Experiment Appendix

In this appendix we provide training/evaluation details alongside additional results.

Information about the different datasets used can be found in the code, where a link to the website of each particular dataset can be found. Information about dataset preprocessing can also be found in the code since datasets are very different (images, tabular, tabular with discrete inputs etc) and so different preprocessing was done. General preprocessing steps are the normalization to $[0, 1]$ range of image datasets, normalization by the mean and standard deviation in continuous tabular datasets, and different normalization procedures depending on the type of feature. For example, a dataset containing working age information was normalized by dividing by 65 since that is the maximum number of years (on average) of a standard worker's life. Also for `absenteeism` we change the task target in order to increase the number of class labels. With `characterfont` we did not use all the available data due to computational constraints. We did this since there is no clear standarization about how these tasks should be evaluated and so we adapted to evaluate the proposed algorithm, which needs high C and a reasonable amount of training data. All this information can be checked in the code's repository.

Common to all experiments is the following information. Experiments are run using GPFLOW (Matthews et al., 2017; van der Wilk et al., 2020). Unless mentioned we use default GPFLOW parameters. Inducing points are initialized using Kmeans algorithm for `vowel`,`absenteeism` and `avila` with 10 reinitializations and parallel Kmeans for `characterfont` and `devangari` with 3 reinitializations. The length scale of RBF kernels was initialized to 2.0 and the mixing matrix randomly. Non-stationary kernels are initialized with a length scale of 2.0 for the arcosine and with an identity matrix for the Neural Network kernel. All kernels employ automatic relevance determination if possible. The variational mean is initialized to zero and the Cholesky factorization of the variational covariance to the identity matrix multiplied by $1e-5$. In all the experiments the model used to compute the train/valid/test metrics was the model corresponding to the epoch with the best (highest) ELBO. We use Adam optimizer with a batch size of 10000. This implies that on the small datasets we are performing full batch gradient descent. This in addition with our deterministic initialization procedure removes most of the randomness in the results, removing the necessity of running several times the same experiments with different seeds. Note that in the ETGP, although the parameters of the NN are initialized randomly, we run an initialization procedure several times removing the influence from the random initialization. On the big datasets we perform stochastic gradient descent however running the models several times to remove possible noise in the stochastic gradient algorithm is unfeasible especially for SVGPs where some of the models took 4 days to run in our computer cluster.

For all the SVGP models we run models with learning rate values of 0.01 and 0.001. For certain choices of hyper-parameters if we saw that 0.01 was providing better results than 0.001 we keep searching just with 0.01. In some cases we also look for other learning rates e.g. 0.05 in light of finding the best baseline model to compare against. We run either 10000 or 15000 epochs for `vowel`,`absenteeism` and `avila` and $100, 200, 500, 1000, 2000$ epochs for `characterfont` and `devangari`. For these last two dataset we do not always launch 2000 epochs, and only did it if we found a big increase in performance from the run with 500 to 1000 epochs. Note that training times are averages over epochs and we do not provide the full time of the experiment (which in turns imply that the ETGP is even faster since we run them just for 500 epochs). We run models with number of inducing points $\{100, 50, 20\}$ for (`vowel`,`absenteeism` and `avila`) and 100 for `characterfont` and `devangari`. We also experiment with the parameters of the covariance (including the mixing matrix parameters in RBFCORR) being frozen for 2000 (`vowel`,`absenteeism` and `avila`) or 50 (`characterfont` and `devangari`) epochs or trained end to end, i.e. no freezing is applied, following Maroñas et al. (2021); Hensman et al. (2015). Once all these experiments were launched, we select for each set of kernel, number of inducing points etc, the model giving the best performance by directly looking at the test set, in order to evaluate the proposed model in the most optimistic situation for each SVGP baseline.

For the ETGP model selection was done using a validation split with a different number of points per dataset. This information is provided by looking at the code that loads the data. For the ETGP all the models are run for 15000 epochs for `vowel`,`absenteeism` and `avila` and 500 epochs for `characterfont` and `devangari` (which implies that the total training time of our models is even faster), and the best selected model on validation for 100 inducing points, is run for 50 and 20, in contrast with SVGP where each 50 and 20 inducing points model can have its own set of training hyperparameters. Bayesian flows are trained with 1 Monte Carlo dropout sample and evaluated (i.e. posterior predictive computation) using 20 dropout samples. The learning rate experimented was 0.01 and 0.001 and all the parameters are trained from the beginning without freezing. The NN architectures were chosen depending on the input size of the dataset. All these architectures have an input layer equal to the dimensionality of the data and an output layer given by the number of parameters of the flow multiplied by the number of classes. We tested LINEAR, SAL (Rios & Tobar, 2019) with length 3 and TANH (Snelson et al.,

2003) with length 3 and 4 elements in the linear combination. The length of the flow corresponds to the value of K in the flow parameterization, i.e. it is the number of, e.g. individual SAL transformations, being concatenated. All the NN use hyperbolic tangent activation function and we use a variance of a Gaussian prior over flow parameters set to $5000, 50000, 50000$ which corresponds to a weight decay factor of $1e-4, 1e-5, 1e-6$ without considering the constant value of the Gaussian prior that depends on the number of parameters. For `vowel`,`absenteeism` and `avila` we test networks with $0, 1, 2$ hidden layers with $25, 50, 100$ neurons per layer and with dropout probabilities of $0.25, 0.5, 0.75$ except `avila` that only uses $0.25, 0.5$. We tested 0.75 to see if higher uncertainty in the NN posterior could help in regularizing the datasets with fewer number of training points. For `devangari` we test $0, 1, 2$ hidden layers with $512, 1024$ neurons per layer. We also tested a projection network of $0, 1$ hidden layers with 512 neurons per hidden layer and 256 neurons per output layer. The output of this projection network is feed into another neural network that maps the 256 dimensions to the number of parameters. This second NN has $0, 1$ hidden layers with $256, 128$ neurons per layer. All these networks have a dropout probability of 0.5. For `characterfont` we also use a dropout probability of 0.5 and NN with $0, 1, 2$ hidden layers with 256 neurons per layer. We also test projection networks of $0, 1, 2$ hidden layers with $512, 256$ neurons per hidden layer and output layer of 256 neurons. This is then feed into another neural network with $0, 1, 2$ hidden layers and 256 neurons per layer.

Regarding the initialization of the flows we follow Maroñas et al. (2021) and initialize the flows to the identity by first learning the identity mapping using a non-input dependent flow, and then learning the parameters of the neural network to match each point in the training dataset to the learned non-input dependent parameters. Both initialization procedures are launched 5 times with a learning rate of 0.05 and Adam optimizer for any dataset and flow architecture. The input-dependent initialization is run for 1000 epochs in `vowel`,`absenteeism` and `avila` and for 100 epochs in `characterfont` and `devangari`. Some preliminary runs were done to test if these hyperparameters allow the flow to be properly initialized and then all these parameters were used for any flow initialization in our validation search without further analysis. We found in general that with fewer epochs the flow could be also initialized properly, but decided to run a considerable number of initialization epochs. We highlight that this procedure can be done in parallel to Kmeans initialization, for readers concerned with the training time associated with this initialization procedure.

B.1. SAL Flow Discussion

In the work from Maroñas et al. (2021) input dependent TGPs only used the SAL flow parameterization. These flows have the good property that they can recover the identity function, see (Rios & Tobar, 2019), making them suitable for these applications since the marginal likelihood can penalize complexity in the warping function and 'choose' to use a GP by setting a linear mapping.

In this work, we have shown that SAL flows provide, in general, worse results than the TANH or LINEAR flows. On one side, this shows the potential improvements that can be achieved by the versatility of the different flows that can be parameterized. Beyond being able to control moments of the induced distributions (Rios & Tobar, 2019), different flows combination can be more expressive or make the training procedure more stable.

In particular, a problem with the SAL flow is that small changes in its parameters can lead to mappings with a high derivative (like an exponential curve). This implies a big change in the gradients and thus while training we can suffer either from numerical saturation or convergence issues. This can be a possible explanation for this performance drop. In fact, in our experiments, we found SAL to be the most saturating flow, i.e. the one making the optimization procedure more unstable.

Nevertheless, exploring architectures with fewer parameters, such as SAL flows, that can learn arbitrary non-linear mappings such as those of TANH, are an interesting line of research. For example, the TANH flow used has 12 parameter, which implies an output NN layer of 1836 neurons for $C = 153$ which considerably increases the computational burden when making Bayesian predictions, and also affects the speed improvement obtained. Anyway, this flow has similar training runs as SVGP while providing much better performance.

B.2. Analysis of `vowel` and `absenteeism` Dataset

For this dataset we observe that SVGP with sharing kernel and inducing points works the best, see Figs. 7,8,3,9. This is because the training and test sets were collected from different speakers pronouncing vowels, and this domain shift can't be captured by these models. As a consequence, the shared K_ν model generalizes better since it under-fits the training set (reflected by worse ACC, LL and ELL in the training set), unlike separate K_ν and ETGP. Since ETGP chooses hyper-parameters using validation data extracted from the training set, this method cannot capture this domain shift. Note that in this particular case, choosing the SVGP with the best test performance is much more beneficial for the SVGP than in other datasets since this

domain shift does not affect the model selected, i.e. the model selected is the one that best captures the domain shift, at the cost of not representing as well the training data. It is expected that if we'd used a validation set to select between SVGP shared and SVGP separate (in a real application), the selected model would be the separate SVGP. Also if the SVGP shared would have been selected using a validation set it is expected that the more expressive one, i.e. the one giving higher training and validation performance (as in ETGP) would likely suffer from domain shift.

As a follow up, we also noted that in some runs the ETGP (mostly those with higher dropout probability i.e. higher epistemic uncertainty) was able to match the results of SVGP with shared covariances. This also supports our claim since when the validation set is close to the training set, the model does not need high epistemic uncertainty to give reliable results on the validation set, and thus a model with low dropout probability is chosen. We expect that techniques that adjust the dropout probability depending on the number of training points will reliably model this dataset since dropout rate is removed from the hyperparameters and its value is fixed depending on the entropy and the number of datapoints (Gal et al., 2017). In any case, our experimental results reflect that epistemic uncertainty is beneficial in domain-shift small datasets, as expected.

We emphasize that this is not a problem of the ETGP but derived from the characteristics of the dataset itself since it is also suffered by SVGP with separate K_ν.

On `absenteeism` (fewer training points than `vowel`) we see that the proposed model (0.307 LINEAR and 0.315 TANH) performs similarly to SVGP (0.314) and correlated SVGP (0.319) in terms of accuracy with separate kernels and inducing points, and better than SVGP with shared kernels and inducing points. This is another sign that domain-shift is responsible for the drop in performance of the more expressive models w.r.t. the SVGP with shared kernels and inducing points in `vowel`. In terms of LL we observe the opposite, being the SVGP with shared kernels and inducing points the one performing best. However, the reason behind this performance is that more expressive models tend to provide higher probabilities towards the wrong class in the errors, increasing the log probability. In other words, the expressive model tends to be more overconfident without sacrificing accuracy, something already observed in Deep Neural Networks (Guo et al., 2017). This is also observed in the ETGP since the Bayesian LINEAR flow (simplest ETGP) performs best on the small datasets `vowel` and `absenteeism`.

B.3. Prediction Time Results

We provide the average prediction time results in Fig. 6, where we can see a similar trend as in the training time results. Our model can be one order of magnitude faster while providing similar or better prediction results. We can see how the reimplementation discussed in App. C boosts the computational performance of the SVGP RBFCORR considerably.

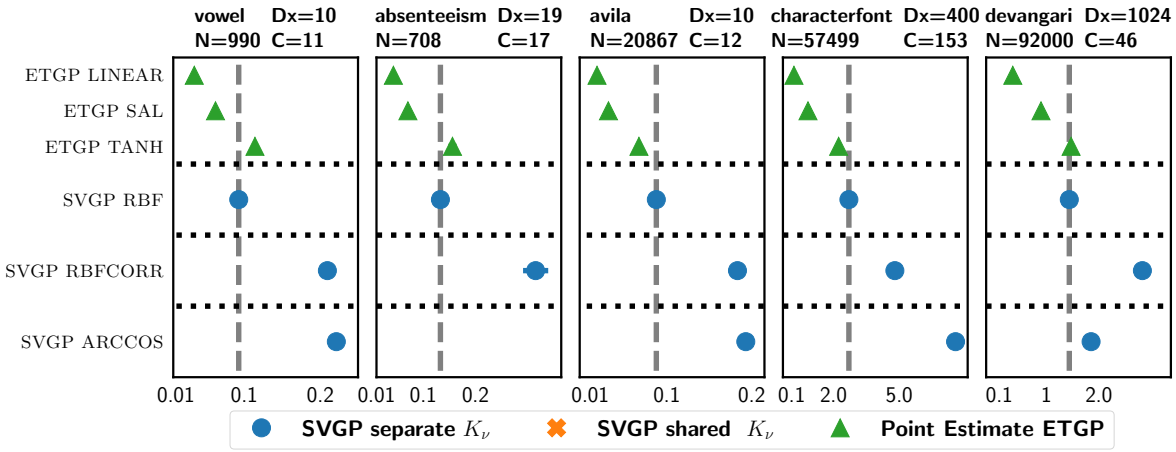

Figure 6. Average prediction time per epoch in minutes (left is better) comparing ETGP with SVGPs. NNET kernel is omitted as it is slower.

B.4. Rest of Results

In this subsection, we provide the missing LL and ACC results. Fig. 7 provides the accuracy comparing ETGP with SVGP with shared kernels and inducing points; Fig. 8 provides the log likelihood comparing ETGP with SVGP with shared kernels and inducing points; Fig. 9 reports log likelihood comparing ETGP with SVGP with separate kernels and inducing points per GP

and Figs. 10,11.

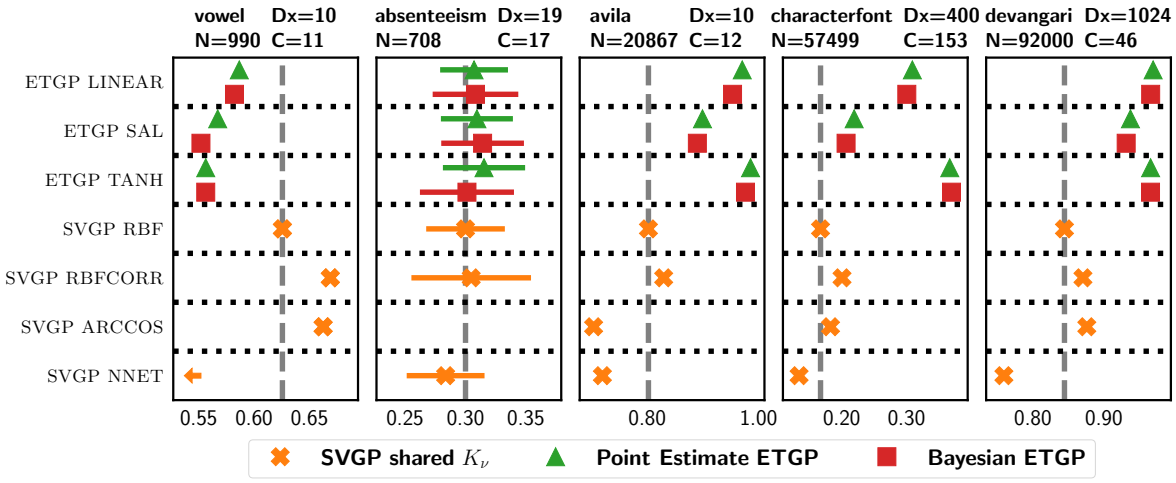

Figure 7. Accuracy comparing ETGP against SVGP with shared kernel and inducing points.

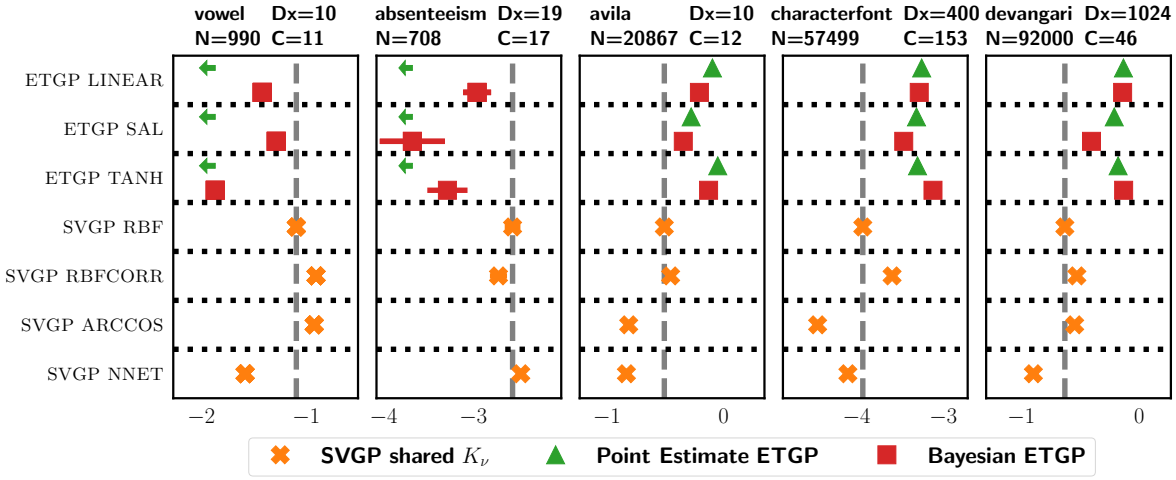

Figure 8. Log Likelihood comparing ETGP against SVGP with shared kernel and inducing points.

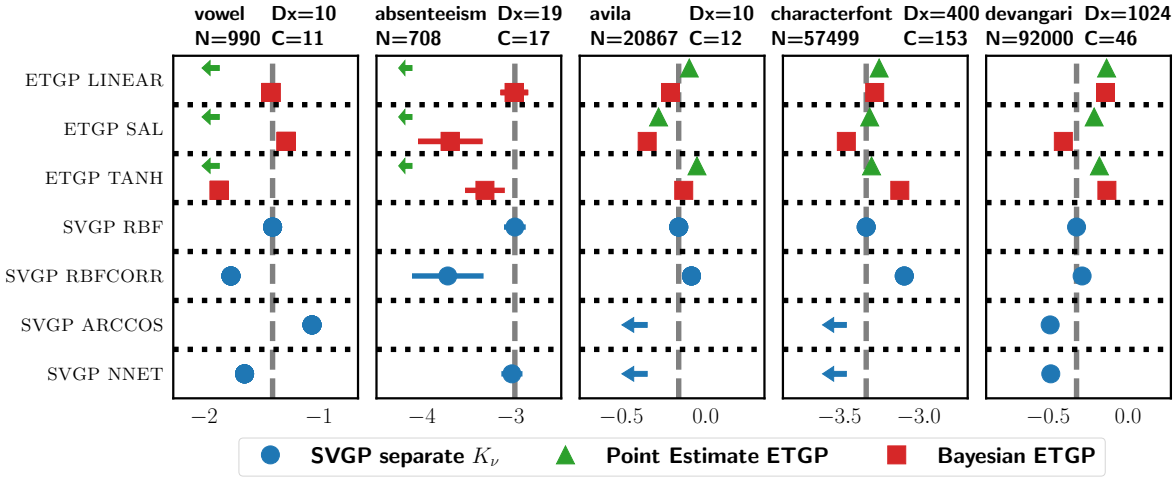

Figure 9. Log Likelihood comparing ETGP against SVGP with separate kernel and inducing points.

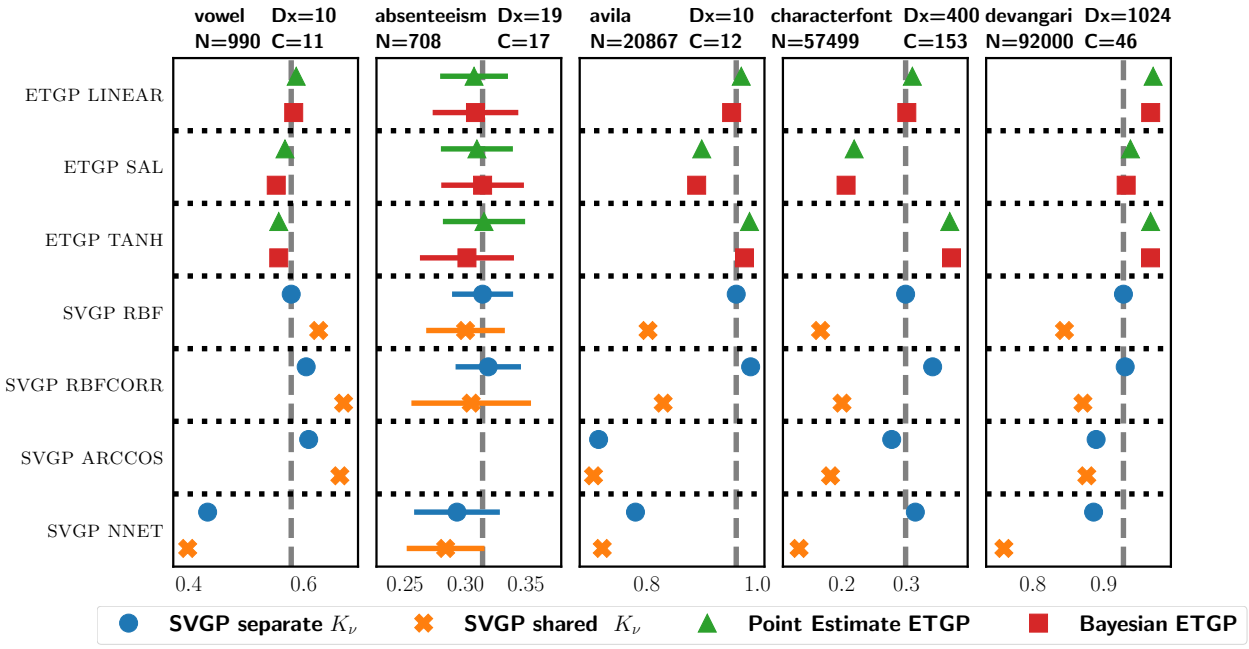

Figure 10. Accuracy of all the models considered in this work. With this figure we can see the relative performance of the different models. Note that the performance increase/decrease might look smaller due to the scale on the x-axis which is increased due to the shared SVGP models. Zoomed versions of this figure are provided in the rest of the work.

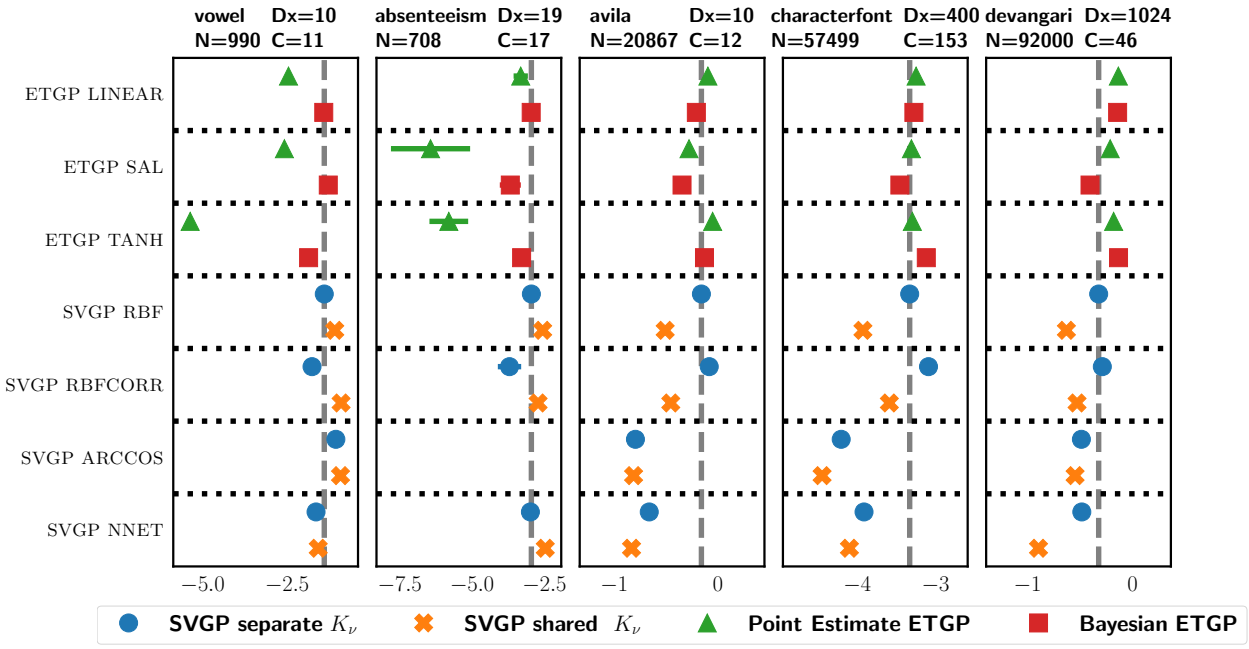

Figure 11. Log Likelihood of all the models considered in this work. With this figure we can see the relative performance of the different models. Note that the performance increase/decrease might look smaller due to the scale on the x-axis which is increased due to the shared SVGP models. Zoomed versions of this figure are provided in the rest of the work.

B.5. Using Less Inducing Points

In Maroñas et al. (2021) it is showed that TGPs could match SVGPs performance in regression problems using 20 times less inducing points. We additionally found that using fewer inducing points can serve also as a regularizer, since the GP posterior is expected to parameterize smoother functions in that case. With this goal, we report results for ETGP using 20, 50 and 100 inducing points, see Figs. 12,13. We observed that ETGP gets regularized when using fewer inducing points regarding ACC. In `vowel` it improves results, and matches or improves SVGPs performance in `avila` and `absenteeism`. We also see how the Bayesian flow provides, in the setting of fewer inducing point, better results than the point estimate. This is especially significant in terms of LL, where we see that all the inducing points provide similar performance.

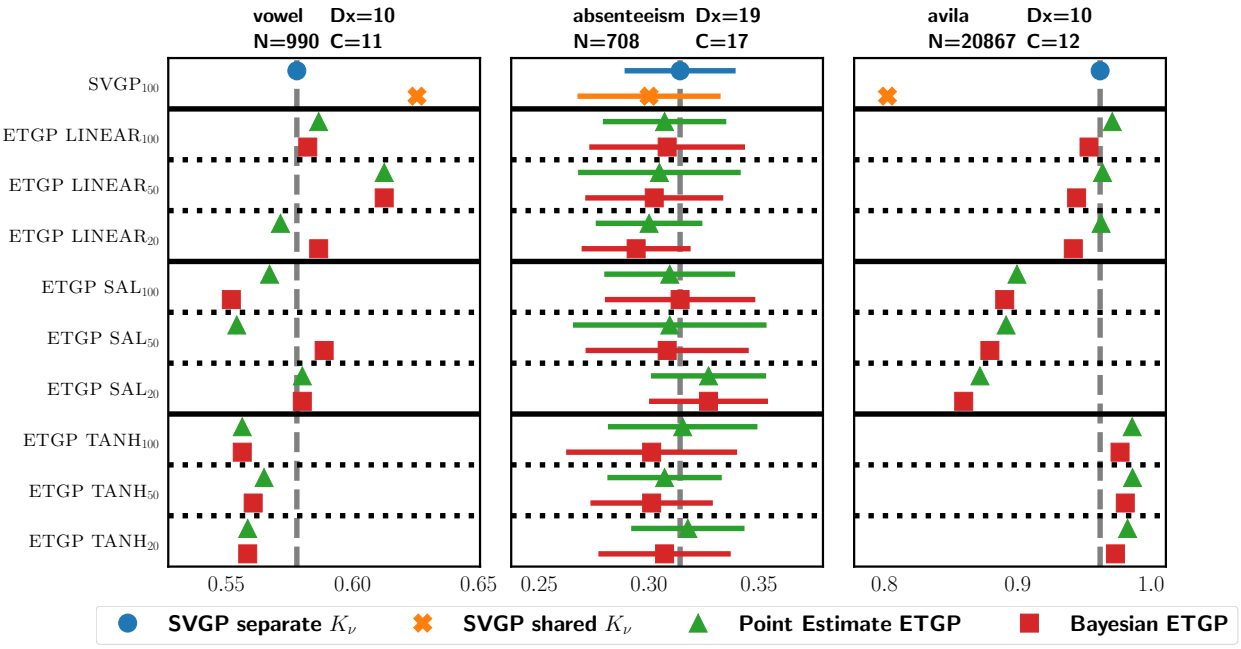

Figure 12. Results comparing ACC (right is better) using less inducing points.

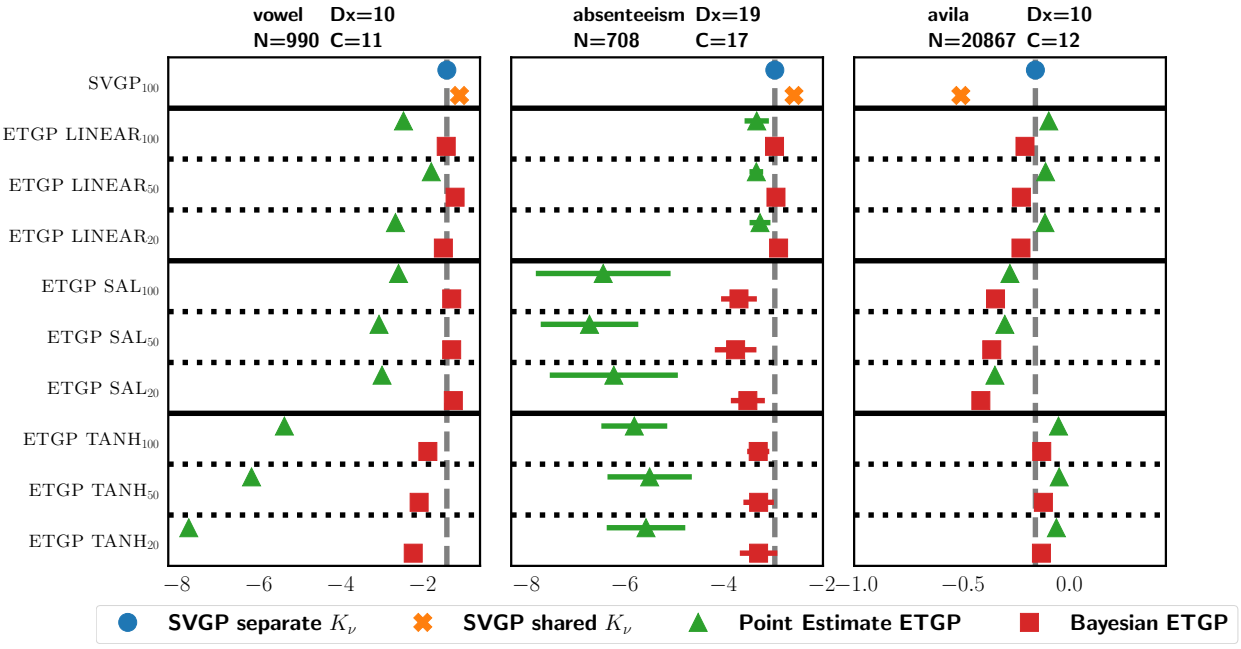

Figure 13. Results comparing LL (right is better) using less inducing points.

B.6. Complete Results of DGP

For the rest of datasets (`vowel` and `absenteeism`) we observe similar results as those with SVGP with shared kernel and inducing points, e.g. DGPS do not suffer from domain shift due to lack of expressiveness. This suggests that even though DGPS with this particular inference algorithm are more expressive than their SVGP counterpart (Salimbeni & Deisenroth, 2017), SVGPS with separate kernels and inducing points are even more expressive. We shall note that most of the experiments in Salimbeni & Deisenroth (2017) require only one GP so there is no way we can compare shared/separate kernels and inducing points. Overall this is an important observation regarding the expressiveness of DGPS, highlighting that the TGP and ETGP family are a good computational and performance alternative to GPS and DGPS.

Table 4. Results for 2 layers DGP

	Accuracy	**Log Likelihood**	**Training Time**
`avila`	0.888	-0.310	0.008
`characterfont`	0.171	-3.636	0.197
`devangari`	0.909	-0.369	0.177
`vowel`	0.658	-0.894	0.0007
`absenteeism`	0.292 ± 0.038	-2.579 ± 0.075	$0.001 \pm 3.278 \cdot 10^{-5}$

C. Refactorizing GPFLOW Source Code

In our experiments we found an issue in the source code of GPFLOW, which considerably speed-up the performance of SVGPs where the kernel is shared. We used GPFLOW version 2.1.3 and have found that the current issue also appears in the last stable version 2.5.2. However, since the source code has considerably changed, our provided implementation is only valid for the earlier version.

We refactorize the source code in a way that is clear where the issue comes from, but note that better refactorizations could be done but would require a high-level and careful check of all the elements involved and how it would generalize to other computations.

The computation is concerned with the evaluation of the marginal variational distribution:

$$q(f_0^c) = \mathcal{N}(f_{0,n}^c \, | K_{\nu\mathbf{x},\mathbf{Z}^c}^c {K_{\nu\mathbf{Z}^c,\mathbf{Z}^c}^c}^{-1} \mathbf{m}^c, \\ K_{\nu\mathbf{x},\mathbf{x}}^c - K_{\nu\mathbf{Z}^c,\mathbf{Z}^c}^c {K_{\nu\mathbf{Z}^c,\mathbf{Z}^c}^c}^{-1} [K_{\nu\mathbf{Z}^c,\mathbf{Z}^c}^c + \mathbf{S}^c] {K_{\nu\mathbf{Z}^c,\mathbf{Z}^c}^c}^{-1} K_{\nu\mathbf{Z}^c,\mathbf{x}}^c)$$

of each of the C latent processes when they are later mixed by a mixing matrix. If we see, for a batch of samples $\mathbf{X}$ we need to evaluate $K_{\nu\mathbf{x},\mathbf{x}}^c$ C times if the kernel is not shared and only 1 if the kernel is shared. We found GPFLOW to always evaluate it C times regardless of the type of kernel when a mixing matrix is later applied. In GPFLOW version 2.1.3 we can see that the LinearCoregionalization kernel does not differentiate if the latent GP is shared or not, see https://github.com/GPflow/GPflow/blob/v2.1.3/gpflow/kernels/multioutput/kernels.py line 174. Thus, when computing the covariance, we can see in line 130 here https://github.com/GPflow/GPflow/blob/v2.1.3/gpflow/conditionals/multioutput/conditionals.py that the evaluation is performed C times when it would be easier to evaluate it just once, and then tile C times, which is the basic refactorization we perform. Also note that if the inducing points are not shared, but the kernel is, GPFLOW would also perform this inefficient computation (see line 129 where the kernel instance is repeated C times). When the inducing points are shared and an independent kernel (no mixing matrix) is used, then GPFLOW runs computations correctly (see lines 46 and 92).

We observed similar issues in the newest GPFLOW version 2.5.2. We see that the LinearCoregionalization kernel in https://github.com/GPflow/GPflow/blob/v2.5.2/gpflow/kernels/multioutput/kernels.py line 199 do not make distinctions on whether the latent kernel is shared or not and this kernel is still inheriting from "Combination" class. Thus, the dispatcher in https://github.com/GPflow/GPflow/blob/v2.5.2/gpflow/posteriors.py will not perform the computation starting in 778 (note it checks if the kernel is SharedIndependent but LinearCoregionalization is type Combination) but that starting in 791, which again evaluates the kernel C times.

We found some improvements that can be done to GPFLOW source code and we plan to open corresponding issues and list them under our Github implementation of this work. We think that this particular one could be easily solved by using a LMC kernel for latent shared kernel and a LMC for latent separate independent kernel; or by explicitly checking if the elements in the underlying kernel list share references; or by, easily, using a list with just one element if the kernel is shared and C elements if the kernel is not shared; or by wrapping around SharedIndependent or SeparateIndependent kernels base classes instead of using a list containing the kernels.

Another option, as employed by GPYTORCH (Gardner et al., 2018), could be to evaluate the kernel directly using batched computations. We believe that the computational bottleneck comes from performing the kernel evaluation using a loop in python. This is because computing the squared distance in most kernels has a linear cost, so at most the corresponding added complexity should be $\mathcal{O}(Cd)$, with d the dimensionality of the data. However, we found that without doing our improvement the code was much slower. Thus we further believe that computing the squared distance using batched computations rather than loops would also bridge the gap between the computational times of the SVGP with a separate kernel and the ETGP. Nevertheless our model is still competitive or even provides better results and faster since cubic computations do not scale linearly with C which in a quick code snippet in TENSORFLOW we found to be 1 order of magnitude slower for $C = 153$ and $M = 100$ and 5 times slower for $C = 10$. The code snippet is:

```python
import tensorflow as tf
import time
import numpy as np

C    = 10
runs = 100

X_ = np.random.randn(1,100,100)
X  = X_@X_.transpose(0,2,1)
X  = tf.constant(X)

Y_ = np.random.randn(C,100,100)
Y  = Y_@Y_.transpose(0,2,1)
Y  = tf.constant(Y)

acc = 0.0
for i in range(runs):
    start = time.process_time();
    tf.linalg.cholesky(X);
    acc += time.process_time()-start
print(acc/runs)

acc = 0.0
for i in range(runs):
    start = time.process_time();
    tf.linalg.cholesky(Y);
    acc += time.process_time()-start
print(acc/runs)

```