Appendix for 'Transforming Gaussian Processes With Normalizing Flows'

Contents

A Mathematical Appendix

In this appendix we provide all the mathematical details and derivations of the model presented in the main paper, plus some additional insights. We start with some definitions that will be used during the appendix. Then we move on the derivation of the ELBO and how we make predictions. The section is ended by strengthening the computational advantages implied by our definition. In order to highlight the role of each of the elements involved in our derivations, we provide a pictorial representation of our model in figure Fig. 1.

A.1 Definitions and Notation

1. We first define some general notation. The finite subset of observations $\{\mathbf{X}^{(n)}\}_{n=1}^{N}$ and inducing points $\{\mathbf{Z}^{(m)}\}_{m=1}^{M}$ from our stochastic process are stacked into matrices $\mathbf{X}$ and $\mathbf{Z}$ respectively. The corresponding function evaluations are stacked into vectors $\mathbf{f}$ and $\mathbf{u}$. We denote with $\mathbf{f}_0$ and $\mathbf{f}_K$ to the function evaluations before and after applying the transformation $\mathbb{G}_\theta$. We denote specific locations n or samples s with additional subscripts e.g. $\mathbf{f}_{0,n,s}$.

2. 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*. However throughout the document we will follow Rezende and Mohamed (2015) and refer to it as *LOTUS rule*. Formally, the above statement implies:

$$\mathbb{E}_{p(\mathbf{f}_K)}[h(\mathbf{f}_K)] = \mathbb{E}_{p(\mathbf{f}_0)}[h(\mathbb{G}_\theta(\mathbf{f}_0))] \tag{1}$$

3. For any transformation $\mathbb{G}$ that induces a valid stochastic process, see Rios (2020) for examples, it holds that the probability distribution at any finite subset of locations $\mathbf{X}$ and $\mathbf{Z}$ is given by:

$$p(\mathbf{f}_K, \mathbf{u}_K) = p(\mathbf{f}_0, \mathbf{u}_0|\mathbf{X}, \mathbf{Z}) \prod_{k=0}^{K-1} \underbrace{\left| \det \begin{pmatrix} \frac{\partial \mathbb{G}_\theta(\mathbf{f}_k)}{\partial \mathbf{f}_k} & \frac{\partial \mathbb{G}_\theta(\mathbf{f}_k)}{\partial \mathbf{u}_k} \\ \frac{\partial \mathbb{G}_\theta(\mathbf{u}_k)}{\partial \mathbf{f}_k} & \frac{\partial \mathbb{G}_\theta(\mathbf{u}_k)}{\partial \mathbf{u}_k} \end{pmatrix} \right|^{-1}}_{\mathbf{J}_{\mathbf{f}_k, \mathbf{u}_k}} \tag{2}$$

Where each element $\frac{\partial \mathbb{G}_\theta(\mathbf{f}_k)}{\partial \mathbf{f}_k}$ is itself the Jacobian of the transformation of function evaluations at $\mathbf{X}$. By noting that the determinant of a block diagonal matrix can be computed as:

$$\det \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \det \left(A - BD^{-1}C\right) \det \left(D\right) \tag{3}$$

We can factorize the joint distribution $p(\mathbf{f}_K, \mathbf{u}_K)$ as follows:

$$\begin{aligned} p(\mathbf{f}_K, \mathbf{u}_K) &= p(\mathbf{f}_K|\mathbf{u}_K)p(\mathbf{u}_K) \\ p(\mathbf{f}_K|\mathbf{u}_K) &= p(\mathbf{f}_0|\mathbf{u}_0) \prod_{k=0}^{K-1} \left| \det \left[\underbrace{\frac{\partial \mathbb{G}_\theta(\mathbf{f}_k)}{\partial \mathbf{f}_k}}_{\mathbf{J}_{\mathbf{f}_k}} - \underbrace{\frac{\partial \mathbb{G}_\theta(\mathbf{f}_k)}{\partial \mathbf{u}_k}}_{\mathbf{J}_{\mathbf{f}_k|\mathbf{u}_k}} \left(\underbrace{\frac{\partial \mathbb{G}_\theta(\mathbf{u}_k)}{\partial \mathbf{u}_k}}_{\mathbf{J}_{\mathbf{u}_k}} \right)^{-1} \underbrace{\frac{\partial \mathbb{G}_\theta(\mathbf{u}_k)}{\partial \mathbf{f}_k}}_{\mathbf{J}_{\mathbf{u}_k|\mathbf{f}_k}} \right] \right|^{-1} \\ p(\mathbf{u}_K) &= p(\mathbf{u}_0) \prod_{k=0}^{K-1} \left| \det \frac{\partial \mathbb{G}_\theta(\mathbf{u}_k)}{\partial \mathbf{u}_k} \right|^{-1} \end{aligned}$$

where we make use of $p(\mathbf{f}_K|\mathbf{u}_K) = {}^{p(\mathbf{f}_K, \mathbf{u}_K)}/_{p(\mathbf{u}_K)}$ to derive the expression for the conditional distribution. Furthermore, note that for marginal flows, B and C are evaluated to zero and we recover the distributions used throughout the inference section in the main paper:

$$\begin{aligned} p(\mathbf{f}_K, \mathbf{u}_K) &= p(\mathbf{f}_K|\mathbf{u}_K)p(\mathbf{u}_K) \\ p(\mathbf{f}_K|\mathbf{u}_K) &= p(\mathbf{f}_0|\mathbf{u}_0) \prod_{k=0}^{K-1} \left| \det \frac{\partial \mathbb{G}_\theta(\mathbf{f}_k)}{\partial \mathbf{f}_k} \right|^{-1} \\ p(\mathbf{u}_K) &= p(\mathbf{u}_0) \prod_{k=0}^{K-1} \left| \det \frac{\partial \mathbb{G}_\theta(\mathbf{u}_k)}{\partial \mathbf{u}_k} \right|^{-1} \end{aligned}$$

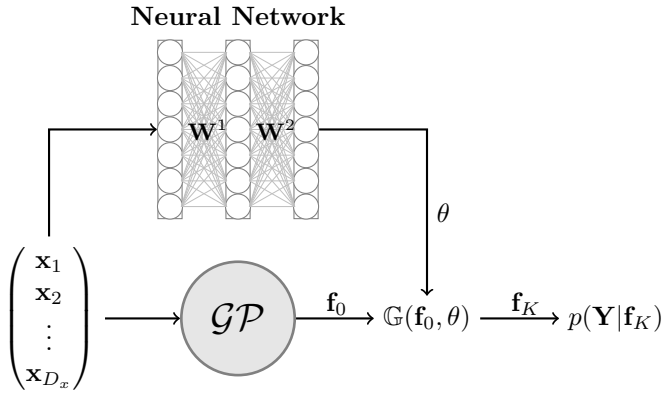


Figure 1: A pictorial representation of our general formulation that highlights the role of the Neural Network. As seen in the figure, the output of the neural network gives the parameters of the flow $\mathbb{G}$. We can further incorporate uncertainty into this parameters by defining a prior $p(\mathbf{W})$ over the NN parameters. See the graphical model in the main article and the details in this appendix.

where now $\frac{\partial \mathbb{G}_\theta(\mathbf{f}_k)}{\partial \mathbf{f}_k}$ is a diagonal matrix where the elements of the diagonal are given by $\frac{\partial \mathbb{G}_\theta(\mathbf{f}_{k,n})}{\partial \mathbf{f}_{k,n}}$.

A.2 Variational Lower Bound Derivation

We now provide a detailed derivation of the variational lower bound of TGP for both the marginal and non-marginal case. In order to reduce the computational cost of directly approximating the posterior at training locations $\mathbf{X}$ (Opper and Archambeau, 2009), we make use of a sparse approximation. We begin by first defining the sparse prior and approximate variational posterior and then derive the respective lower bounds for marginal and non-marginal flows. As we will see the only difference between marginal and non marginal flows is that in the later the inducing points cannot be integrated with analytically, and so we resort to a Monte Carlo approximation.

A.2.1 Sparse Prior

For computational efficiency we follow Titsias (2009); Hensman et al. (2013) and augment the TGP prior with inducing points $\mathbf{u}_K$ at inducing locations $\mathbf{Z}$. The sparse prior of TGP is defined by:

$$p(\mathbf{f}_K, \mathbf{u}_K) = p(\mathbf{f}_K \mid \mathbf{u}_K)p(\mathbf{u}_K) = p(\mathbf{f}_0 \mid \mathbf{u}_0)p(\mathbf{u}_0)\mathbf{J}_{\mathbf{f}_K, \mathbf{u}_K} \tag{4}$$

where the $p(\mathbf{u}_0)$ is a GP and the conditional $p(\mathbf{f}_0 \mid \mathbf{u}_0)$ is a standard Gaussian conditional:

$$\begin{aligned} p(\mathbf{u}_0) &= \mathcal{N}(\mathbf{u}_0 \mid 0, K_{\mathbf{Z},\mathbf{Z}}) \\ p(\mathbf{f}_0 \mid \mathbf{u}_0) &= \mathcal{N}(\mathbf{f}_0 \mid K_{\mathbf{X},\mathbf{Z}}K_{\mathbf{Z},\mathbf{Z}}^{-1}\mathbf{u}_0, K_{\mathbf{X},\mathbf{X}} - K_{\mathbf{X},\mathbf{Z}}K_{\mathbf{Z},\mathbf{Z}}^{-1}K_{\mathbf{Z},\mathbf{X}}) \end{aligned} \tag{5}$$

and K is positive-semi definite kernel function. As described in the main paper, to sparsify this prior the transformation $\mathbb{G}$ must induce a valid stochastic process. It holds that marginal flows always induce a valid stochastic process, but some care has to be taken in the definition of non-marginal flows. We refer the reader to (Rios, 2020) for examples of non-marginal transformations $\mathbb{G}$ that transforms $p(\mathbf{f}_0)$ in a consistent way.

A.2.2 Approximate posterior

Following Titsias (2009) we define our approximate posterior such that the conditional terms cancel:

$$q(\mathbf{f}_K \mid \mathbf{u}_K) = p(\mathbf{f}_K \mid \mathbf{u}_K)q(\mathbf{u}_K) \tag{6}$$

where

$$q(\mathbf{u}_K) = \mathcal{N}(\mathbf{u}_0 \mid \mathbf{m}, \mathbf{S})\mathbf{J}_{\mathbf{u}_K} \tag{7}$$

with $\mathbf{m} \in \mathbb{R}^{M\times 1}$ and $\mathbf{S} \in \mathbb{R}^{M\times M}$ being the variational parameters and $p(\mathbf{f}_K|\mathbf{u}_K)$ is given by definition 3.

A.2.3 ELBO

Following a similar derivation as Hensman et al. (2013) & Blei et al. (2017), we can arrive at the following ELBO:

$$\begin{aligned}\text{ELBO} &= \mathbb{E}_{q(\mathbf{f}_K, \mathbf{u}_K)}\left[\log \frac{\prod_n p(\mathbf{Y}_n \mid \mathbf{f}_{K,n}) p(\mathbf{f}_K, \mathbf{u}_K)}{q(\mathbf{f}_K, \mathbf{u}_K)}\right] \\ &= \underbrace{\sum_n^N \mathbb{E}_{q(\mathbf{f}_K, \mathbf{u}_K)}\left[\log p(\mathbf{Y}_n \mid \mathbf{f}_{K,n})\right]}_{\text{ELL}} + \underbrace{\mathbb{E}_{q(\mathbf{f}_K, \mathbf{u}_K)}\left[\log \frac{p(\mathbf{f}_K, \mathbf{u}_K)}{q(\mathbf{f}_K, \mathbf{u}_K)}\right]}_{-\text{KL}}\end{aligned} \tag{8}$$

In order to highlight the difference between using marginal and non-marginal flows, we break the derivation of the final lower bound into the KL and the expected log likelihood (ELL). As we will illustrate, only the ELL depends on the type of transformation (marginal or non) and so we first deal with the (negative) KL term.

A.2.4 KL divergence

It turns out that the KL between the prior and the approximate posterior on the transformed space $\mathbf{f}_K$, is given by the same KL on the original space $\mathbf{f}_0$:

$$\begin{aligned}\text{KL} &= -\mathbb{E}_{q(\mathbf{f}_K, \mathbf{u}_K)}\left[\log \frac{\cancel{p(\mathbf{f}_K \mid \mathbf{u}_K)} p(\mathbf{u}_0) \cancel{\mathbf{J}_{\mathbf{u}_K}}}{\cancel{p(\mathbf{f}_K \mid \mathbf{u}_K)} q(\mathbf{u}_0) \cancel{\mathbf{J}_{\mathbf{u}_K}}}\right] \\ &= -\mathbb{E}_{q(\mathbf{u}_K)}\left[\log \frac{p(\mathbf{u}_0)}{q(\mathbf{u}_0)}\right] \\ &= \underbrace{-\mathbb{E}_{q(\mathbf{u}_0)}\left[\log \frac{p(\mathbf{u}_0)}{q(\mathbf{u}_0)}\right]}_{:=\text{KL}[q(\mathbf{u}_0)||p(\mathbf{u}_0)]}\end{aligned} \tag{9}$$

where we have first marginalized $\mathbf{f}_K$ and then applied the LOTUS rule. By defining the approximate posterior as being transformed by the same flow as the prior it allows both the Jacobian and conditional distribution $p(\mathbf{f}_K|\mathbf{u}_K)$ to cancel. This not only alleviates costly $\mathcal{O}(N^3)$ computations coming from the KL between the conditional distributions, but also simplifies the KL to simply be between two Gaussian distributions.

A.2.5 ELL with Marginal Flows

We now derive the expected log term for the case in which $\mathbb{G}$ is a marginal flow. To do so we first derive the form of $q(\mathbf{f}_K)$ by analytically integrating out the inducing points $\mathbf{u}_K$. We then marginalize $\mathbf{f}_K$ such that the ELL term decomposes into components of $\mathbf{f}_{K,n}$ and $\mathbf{Y}_n$. The marginal $q(\mathbf{f}_K)$ is given by:

$$\begin{aligned}q(\mathbf{f}_K) &= \int q(\mathbf{f}_K, \mathbf{u}_K) d\mathbf{u}_K \\ &= \int p(\mathbf{f}_0|\mathbf{u}_0) q(\mathbf{u}_0) \mathbf{J}_{\mathbf{f}_K, \mathbf{u}_K} d\mathbf{u}_K\end{aligned} \tag{10}$$

Because $\mathbb{G}$ is a marginal flow the Jacobian matrix is diagonal and decomposes such that $\mathbf{J}_{\mathbf{f}_K, \mathbf{u}_K} = \mathbf{J}_{\mathbf{f}_K} \mathbf{J}_{\mathbf{u}_K}$. Substituting this into the above and applying LOTUS rule by recognizing this as an expectation w.r.t $q(\mathbf{u}_K)$:

$$\begin{aligned}q(\mathbf{f}_K) &= \int p(\mathbf{f}_0|\mathbf{u}_0) q(\mathbf{u}_0) \mathbf{J}_{\mathbf{f}_K} \mathbf{J}_{\mathbf{u}_K} d\mathbf{u}_K \\ &= \mathbf{J}_{\mathbf{f}_K} \int p(\mathbf{f}_0|\mathbf{u}_0) q(\mathbf{u}_0) d\mathbf{u}_0 \\ &= \mathbf{J}_{\mathbf{f}_K} q(\mathbf{f}_0)\end{aligned} \tag{11}$$

where $\mathbf{J}_{\mathbf{f}_K}$ does not depend on $\mathbf{u}_0$ and the marginal $q(\mathbf{f}_0)$ is:

$$q(\mathbf{f}_0) = \mathcal{N}(\mathbf{f}_0 \mid K_{\mathbf{X},\mathbf{Z}} K_{\mathbf{Z},\mathbf{Z}}^{-1} \mathbf{m}, K_{\mathbf{X},\mathbf{X}} - K_{\mathbf{X},\mathbf{Z}} K_{\mathbf{Z},\mathbf{Z}}^{-1} \left[K_{\mathbf{Z},\mathbf{Z}} + \mathbf{S}\right] K_{\mathbf{Z},\mathbf{Z}}^{-1} K_{\mathbf{Z},\mathbf{X}}) \tag{12}$$

Making use of this derivation of $q(\mathbf{f}_K)$ we can simplify the ELL term in Eq. 8:

$$\begin{aligned}\text{ELL} &= \sum_n^N \mathbb{E}_{q(\mathbf{f}_K, \mathbf{u}_K)} \left[\log p(\mathbf{Y}_n \mid \mathbf{f}_{K,n})\right]] \\ &= \sum_n \mathbb{E}_{q(\mathbf{f}_0)} \left[\log p(\mathbf{Y}_n \mid \mathbf{f}_{K,n})\right] \\ &= \sum_n \mathbb{E}_{q(\mathbf{f}_{0,n})} \left[\log p(\mathbf{Y}_n \mid \mathbf{f}_{K,n})\right],\end{aligned} \tag{13}$$

where we have now applied LOTUS rule over the expectation w.r.t $q(\mathbf{f}_K)$ after integrating out $\mathbf{u}_K$. We finally integrate out all the elements $\mathbf{f}_0$ but $\mathbf{f}_{0,n}$ from our variational posterior by noting that $\mathbf{Y}_n$ only depends on the function evaluation at position n. Thus, the ELL term now factorizes across $\mathbf{Y}_n$ and the latent variables $\mathbf{f}_{0,n}$, as required for SVI. Plugging this into the ELBO:

$$\text{ELBO} = \sum_n^N \mathbb{E}_{q(\mathbf{f}_{0,n})} \left[\log p(\mathbf{Y}_n \mid \mathbb{G}(\mathbf{f}_{0,n}))\right] + \mathbb{E}_{q(\mathbf{u}_0)} \left[\log \frac{p(\mathbf{u}_0)}{q(\mathbf{u}_0)}\right] \tag{14}$$

recovers the lower bound presented in the main paper.

A.2.6 ELL with Non Marginal Flows

We now present a generalization of the presented inference algorithm to include non-marginal flows where, as before, we only require that $\mathbb{G}$ induces a valid stochastic process. The key difference between using marginal and non marginal flows is that for non-marginal flows we will not be able to, in general, analytically integrate out the inducing points $\mathbf{u}_K$. However, we can simply integrate them with Monte Carlo. To illustrate this fact, we proceed as in the previous section. The marginal $q(\mathbf{f}_K)$ is given by:

$$\begin{aligned}q(\mathbf{f}_K) &= \int q(\mathbf{f}_K, \mathbf{u}_K) d\mathbf{u}_K \\ &= \int p(\mathbf{f}_K|\mathbf{u}_K) q(\mathbf{u}_K) d\mathbf{u}_K \\ &= \int [\mathbf{J}_{\mathbf{f}_K} - \mathbf{J}_{\mathbf{f}_K|\mathbf{u}_K} \mathbf{J}_{\mathbf{u}_K}^{-1} \mathbf{J}_{\mathbf{u}_K|\mathbf{f}_K}] p(\mathbf{f}_0|\mathbf{u}_0) q(\mathbf{u}_0) d\mathbf{u}_0 \\ &= q(\mathbf{f}_0)\mathbf{J}_{\mathbf{f}_K} - \int [\mathbf{J}_{\mathbf{f}_K|\mathbf{u}_K} \mathbf{J}_{\mathbf{u}_K}^{-1} \mathbf{J}_{\mathbf{u}_K|\mathbf{f}_K}] p(\mathbf{f}_0|\mathbf{u}_0) q(\mathbf{u}_0) d\mathbf{u}_0\end{aligned} \tag{15}$$

where in the last line we use LOTUS rule. Note that integrating the inducing points will be generally intractable due to the non-linearity of the flow $\mathbb{G}$ that appears in the conditional prior $p(\mathbf{f}_K|\mathbf{u}_K)$ through the elements $\mathbf{J}_{\mathbf{f}_K|\mathbf{u}_K}$,$\mathbf{J}_{\mathbf{u}_K|\mathbf{f}_K}$ and $\mathbf{J}_{\mathbf{u}_K}$. Resorting to a Monte-Carlo approximation and simplifying the ELL term in Eq. 8:

$$\begin{aligned}\text{ELL} &= \sum_n \mathbb{E}_{q(\mathbf{f}_K, \mathbf{u}_K)} \left[\log p(\mathbf{Y}_n \mid \mathbf{f}_{K,n})\right] \\ &= \sum_n \mathbb{E}_{q(\mathbf{f}_{K,n}, \mathbf{u}_K)} \left[\log p(\mathbf{Y}_n \mid \mathbf{f}_{K,n})\right] \\ &\approx \sum_n \frac{1}{S} \sum_s \left[\log p(\mathbf{Y}_n \mid \mathbf{f}_{K,n,s})\right]\end{aligned}$$

where we follow similar steps to marginal flows, i.e. we integrate out all the elements from $\mathbf{f}_K$ but $\mathbf{f}_{K,n}$ (see below section). The last line is the Monte-Carlo approximation where samples are obtained by the generative process defined for flow based models, i.e *sample from the base distribution* and *warp samples with the flow*:

$$\begin{aligned}&\mathbf{u}_{0,s} \sim q(\mathbf{u}_0) \\ &\mathbf{f}_{0,n,s} \sim p(\mathbf{f}_{0,n}|\mathbf{u}_{0,s}) \\ &\mathbf{f}_{K,n,s}, \mathbf{u}_{K,s} = \mathbb{G}_\theta(\mathbf{f}_{0,n,s}, \mathbf{u}_{0,s})\end{aligned} \tag{16}$$

where the samples $\mathbf{u}_{K,s}$ are then discarded. Evaluation of this ELBO requires a computational complexity of $\mathcal{O}(M^3)$ and although it requires two layers of sampling (from $q(\mathbf{u}_0)$ and $p(\mathbf{f}_{0,n} \mid \mathbf{u}_0)$) this is similar to the

doubly stochastic framework of Salimbeni and Deisenroth (2017) where samples must be propagated through the layers of the DGP. Both sampling distributions are Gaussian and so the reparameterization trick can be used to generate low variance, unbiased gradients (Kingma and Welling, 2014). Moreover, because $p(\mathbf{f}_{0,n} \mid \mathbf{u}_0)$ is a univariate Gaussian sampling can be easily parallelized. No matter the choice of $\mathbb{G}$ we have shown that we can always factorize the ELL term across $\mathbf{Y}$ and $\mathbf{f}_K$ making this bound applicable to stochastic variational inference (Hensman et al., 2013). This allows the ELL term to be approximated through mini-batching. Although this results in noisy gradient updates it allows TGP to be trained on millions of observations.

A.2.7 Computing the marginal distribution

We now explicitly derive the marginal $q(\mathbf{f}_{K,n})$ by integrating $\mathbf{f}_{K,a\neq n}$ from $q(\mathbf{f}_K)$ from an alternative perspective. The derivation rests on a similar assumption made by Titsias (2009) where $\mathbf{u}_K$ is assumed to be sufficient statistics for $\mathbf{f}_K$. Expanding the marginal $q(\mathbf{f}_{K,n})$:

$$\begin{aligned} q(\mathbf{f}_{K,n}) &= \int q(\mathbf{f}_{K,n}, \mathbf{f}_{K,a}) \mathrm{d}\mathbf{f}_{K,a} \\ &= \int p(\mathbf{f}_{K,n}, \mathbf{f}_{K,a} \mid \mathbf{u}_K) q(\mathbf{u}_K) \mathrm{d}\mathbf{f}_{K,a} \mathrm{d}\mathbf{u}_K \\ &= \int p(\mathbf{f}_{K,n} \mid \mathbf{u}_K, \mathbf{f}_{K,a}) p(\mathbf{f}_{K,a} \mid \mathbf{u}_K) q(\mathbf{u}_K) \mathrm{d}\mathbf{f}_{K,a} \mathrm{d}\mathbf{u}_K \\ &= \int p(\mathbf{f}_{K,n} \mid \mathbf{u}_K) q(\mathbf{u}_K) \mathrm{d}\mathbf{u}_K \end{aligned} \tag{17}$$

For marginal flows this integration can be done analytically but for non-marginal flows we resort to Monte-Carlo approximations. The important point here is that no matter the form of $\mathbb{G}$ we just need the finite dimensional distributions at $\mathbf{Z}$ and $\mathbf{X}^{(n)}$ to evaluate the lower bound.

A.3 Bayesian Input-Dependent Flows

To finish with the mathematical derivations that complete the full specification of our variational posterior, we describe the necessary steps for marginal Bayesian input-dependent flows. The core idea is that the parameters of the flows that apply on each of the function evaluations $\mathbf{f}_{0,n}$ at each index $\mathbf{X}^{(n)}$ of the stochastic process, depend directly on the index $\mathbf{X}^{(n)}$ – rather than being shared across them. Note that as the flow still applies over each of the function evaluations $\mathbf{f}_{0,n}$ independently, then our input-dependent flows correspond to an input-dependent marginal flow. For arbitrary flows, one has to make sure that the necessary conditions are satisfied when the flow is made input-dependent. Hence this subsection just illustrates the derivation of the ELBO within the derivation done for marginal flows.

To do so, we assume independence between the stochastic process and the parameters of the Neural Network both on the prior and the posterior:

$$\begin{aligned} p(\mathbf{f}_K, \mathbf{u}_K, \mathbf{W}) &= p(\mathbf{f}_K | \mathbf{u}_K) p(\mathbf{u}_K) p(\mathbf{W}) \\ q(\mathbf{f}_K, \mathbf{u}_K, \mathbf{W}) &= p(\mathbf{f}_K | \mathbf{u}_K) q(\mathbf{u}_K) q_\phi(\mathbf{W}) \end{aligned} \tag{18}$$

where ϕ are the variational parameters of the BNN. By plugging this into the ELBO we arrive at:

$$\begin{aligned} \text{ELBO} &= \mathbb{E}_{q(\mathbf{f}_K, \mathbf{u}_K) q(\mathbf{W})} \log \left[\frac{\prod_n p(\mathbf{Y}_n | \mathbf{f}_{K,n}) p(\mathbf{f}_K, \mathbf{u}_K) p(\mathbf{W})}{q(\mathbf{f}_K, \mathbf{u}_K) q_\phi(\mathbf{W})} \right] \\ &= \mathbb{E}_{q(\mathbf{f}_0) q_\phi(\mathbf{W})} \log \left[\prod_n p(\mathbf{Y}_n | \mathbf{f}_{K,n}) \right] - \mathrm{KL}[q(\mathbf{u}_0) || p(\mathbf{u}_0) - \mathrm{KL}[q_\phi(\mathbf{W}) || p(\mathbf{W}) \\ &= \sum_n \mathbb{E}_{q(\mathbf{f}_{0,n}) q_\phi(\mathbf{W})} \log \left[p(\mathbf{Y}_n | \mathbb{G}_{\theta(\mathbf{X}, \mathbf{W})}(\mathbf{f}_{0,n})) \right] - \mathrm{KL}[q(\mathbf{u}_0) || p(\mathbf{u}_0)] - \mathrm{KL}[q_\phi(\mathbf{W}) || p(\mathbf{W})] \\ &\approx \sum_n \frac{1}{S} \sum_s \mathbb{E}_{q(\mathbf{f}_{0,n})} \log \left[p(\mathbf{Y}_n | \mathbb{G}_{\theta(\mathbf{X}, \mathbf{W}_s)}(\mathbf{f}_{0,n})) \right] - \mathrm{KL}[q(\mathbf{u}_0) || p(\mathbf{u}_0)] - \mathrm{KL}[q_\phi(\mathbf{W}) || p(\mathbf{W})] \end{aligned}$$

where $W_s \sim q_\phi(\mathbf{W})$. This bound has two interesting properties. First one can allow for low variance and unbiased gradients w.r.t ϕ by reparameterization (something satisfied for popular choices of $q(\mathbf{W})$ such as the mean-field Gaussian family). Second, one can account for prior miss-specification by substituting the KL for other divergences, which has been shown to improve the performance of this model (see Knoblauch et al., 2019).

In our work however we have implemented the BNN using Monte Carlo dropout (Gal and Ghahramani, 2016) as it can be more efficiently trained and also allow us to avoid some well-known problems of mean field VI such as variance under-estimation see e.g Bishop (2006). Nevertheless, our bound can be efficiently trained regardless of the specification of the variational family by using batched matrix computations.

Finally, note that computing the forward passes through the Neural Network are independent of the computation of $q(\mathbf{f}_0)$. This means that one can parallelize the computation of $q(\mathbf{f}_0)$ and $\theta(\mathbf{W}_s, \mathbf{X})$.

A.4 Predictions

To make predictions we replace the true, unknown, posterior with the variational approximation. At a test location $\mathbf{X}^*$, after integrating out the inducing points, we have:

$$\begin{aligned} p(\mathbf{Y}^*|\mathbf{X}^*, \mathbf{X}, \mathbf{Y}) &= \int p(\mathbf{Y}^*|\mathbf{f}_K^*) p(\mathbf{f}_K^*|\mathbf{X}, \mathbf{Y}) d\mathbf{f}_K^* \approx \\ &\approx \int p(\mathbf{Y}^*|\mathbf{f}_K^*) q(\mathbf{f}_K^*) d\mathbf{f}_K^* \\ &\approx \int p(\mathbf{Y}^*|\mathbf{f}_K^*) q(\mathbf{f}_0^*) d\mathbf{f}_0^* \end{aligned} \tag{19}$$

where we have again make use of the LOTUS rule. This integral can be computed with quadrature. Note that for arbitrary flows, one can integrate the inducing points by the same sampling procedure we have already introduced in Sec. A.2.6. For the case of Bayesian input-dependent flows, we further approximate the integral using Monte Carlo:

$$\begin{aligned} p(\mathbf{Y}^*|\mathbf{X}^*, \mathbf{X}, \mathbf{Y}) &= \int p(\mathbf{Y}^*|\mathbf{f}_K^*, \theta(\mathbf{W}, \mathbf{X}^*)) p(\mathbf{f}_K^*, \mathbf{W}|\mathbf{X}, \mathbf{Y}) d\mathbf{f}_K^* \approx \\ &\approx \int p(\mathbf{Y}^*|\mathbf{f}_K^*) q(\mathbf{f}_K^*) q(\mathbf{W}) d\mathbf{f}_K^* d\mathbf{W} \\ &\approx \frac{1}{S} \sum_s \int p(\mathbf{Y}^*|\mathbf{f}_K^*, \theta(\mathbf{W}_s, \mathbf{X}^*)) q(\mathbf{f}_0^*) d\mathbf{f}_0^* \end{aligned} \tag{20}$$

where again each element of the Monte Carlo is a 1 dimensional quadrature integral. The whole process for both input and non-input-dependent flows can be easily parallelized with batched operations over matrices. Confidence intervals are obtained by sampling from the approximate posterior predictive and computing the relevant quantiles. Moment estimation and predictive test log likelihood is done by one dimensional quadrature, Monte Carlo estimation, and the logsumexp trick. We provide full derivations of the necessary estimators in our GitHub implementation.

A.4.1 Predicting with V-WGP

The confidence intervals of the predictive distribution with a transformed likelihood ($\mathbb{T} \neq \mathbf{I}$)) are the transformed confidence intervals of $p(\mathbb{T}(\mathbf{Y}^*))$:

$$CI(\mathbf{Y}^*) = \mathbb{T}^{-1}(CI(T(\mathbf{Y}^*)))$$

Prediction with a transformed likelihood requires evaluating the inverse of $\mathbb{T}$ which in general requires approximation such as Newton-Raphson.

A.5 Alternative Variational Families

A.5.1 Derivation of G-SP

We use G-SP to denote the model which uses a Gaussian variational family instead of the transformed Gaussian proposed in the main paper. The joint model is the same as the TGP:

$$p(\mathbf{Y}, \mathbf{f}_K, \mathbf{u}_K) = p(\mathbf{Y} \mid \mathbf{f}_K) p(\mathbf{f}_K, \mathbf{u}_K) \tag{21}$$

where $p(\mathbf{f}_K, \mathbf{u}_K)$ is given in Eq. 4 as :

$$p(\mathbf{f}_K, \mathbf{u}_K) = p(\mathbf{f}_0 \mid \mathbf{u}_0) p(\mathbf{u}_0) \mathbf{J}_{\mathbf{f}_K, \mathbf{u}_K} \tag{22}$$

with $\mathbf{J}_{\mathbf{f}_K, \mathbf{u}_K} = \left| \det \frac{\partial \mathbb{G}_\theta^{-1}(\mathbf{f}_K)}{\partial \mathbf{f}_K} \right| \left| \det \frac{\partial \mathbb{G}_\theta^{-1}(\mathbf{u}_K)}{\partial \mathbf{u}_K} \right|$. The derivation of the variational follows that of the SVGP and TGP. Let $q(\mathbf{f}_K, \mathbf{u}_K) = p(\mathbf{f}_K \mid \mathbf{u}_K) q(\mathbf{u}_K)$ be the Gaussian approximate posterior such that:

$$\begin{aligned} q(\mathbf{u}_K) &= \mathcal{N}(\mathbf{u}_K \mid \mathbf{m}, \mathbf{S}) \\ p(\mathbf{f}_K \mid \mathbf{u}_K) &= \mathcal{N}(\mathbf{f}_K \mid K_{\mathbf{X},\mathbf{Z}} K_{\mathbf{Z},\mathbf{Z}}^{-1} \mathbf{u}_K, K_{\mathbf{X},\mathbf{X}} - K_{\mathbf{X},\mathbf{Z}} K_{\mathbf{Z},\mathbf{Z}}^{-1} K_{\mathbf{Z},\mathbf{X}}) \end{aligned} \tag{23}$$

then the ELBO is given by:

$$\begin{aligned} \mathcal{L}_{\text{G-SP}} &= \mathbb{E}_{q(\mathbf{f}_K, \mathbf{u}_K)} \left[\log \frac{p(\mathbf{Y} \mid \mathbf{f}_K) p(\mathbf{f}_0 \mid \mathbf{u}_0) p(\mathbf{u}_0) \mathbf{J}_{\mathbf{f}_K, \mathbf{u}_K}}{p(\mathbf{f}_K \mid \mathbf{u}_K) q(\mathbf{u}_K)} \right] \\ &= \mathbb{E}_{q(\mathbf{f}_K)} \left[\log p(\mathbf{Y} \mid \mathbf{f}_K) \right] + \mathbb{E}_{q(\mathbf{f}_K, \mathbf{u}_K)} \left[\log \frac{p(\mathbf{f}_0 \mid \mathbf{u}_0)}{p(\mathbf{f}_K \mid \mathbf{u}_K)} \right] + \mathbb{E}_{q(\mathbf{f}_K, \mathbf{u}_K)} \left[\log \frac{p(\mathbf{u}_0)}{q(\mathbf{u}_K)} \right] + \mathbb{E}_{q(\mathbf{f}_K, \mathbf{u}_K)} \left[\log \mathbf{J}_{\mathbf{f}_K, \mathbf{u}_K} \right] \\ &= \mathbb{E}_{q(\mathbf{f}_K)} \left[\log p(\mathbf{Y} \mid \mathbf{f}_K) \right] - \mathbb{E}_{q(\mathbf{u})} \left[\mathrm{KL} \left[p(\mathbf{f}_K \mid \mathbf{u}_K) \mid\mid p(\mathbf{f}_0 \mid \mathbf{u}_0) \right] \right] - \mathrm{KL} \left[q(\mathbf{u}_K) \mid\mid p(\mathbf{u}_0) \right] + \mathbb{E}_{q(\mathbf{f}_K, \mathbf{u}_K)} \left[\log \mathbf{J}_{\mathbf{f}_K, \mathbf{u}_K} \right] \end{aligned} \tag{24}$$

The approximate posterior is no longer transformed by the same flow as the prior which means the Gaussian conditionals and Jacobian term no longer cancel. The first term is the same expected log likelihood term found in the SVGP and so can be computed in closed form or with one dimensional quadrature as required. The second and fourth term must be approximated using monte-carlo due to the dimension of the base distribution.

Prediction using G-SP follows the standard SVGP:

$$\begin{aligned} p(\mathbf{Y}^*) &= \int p(\mathbf{Y}^* \mid \mathbf{f}_K^*) p(\mathbf{f}_K^*) \, \mathrm{d}\mathbf{f}_K^* \\ &\approx \int p(\mathbf{Y}^* \mid \mathbf{f}_K^*) q(\mathbf{f}_K^*) \, \mathrm{d}\mathbf{f}_K^* \end{aligned} \tag{25}$$

which can be computed using the SVGP prediction equations.

A.6 Other Computational Aspects

To end this part of the appendix, we highlight and summarize some convenient computational aspects that derive from our approximation and what other modeling choices imply.

One of the most interesting properties of our model is that both training and predictions can be done without inverting $\mathbb{G}$ – which allow us to use any expressive invertible transformation (see App. B). This contrasts with models that warp the likelihood, where either strong constraints must be placed on the kind of transformations employed so that the inverse can be computed analytically (Rios and Tobar, 2019); or the inverse has to be computed using numerical methods as Newton-Raphson (Snelson et al., 2003).

On the other hand, other choices for the variational distribution would imply an undesirable increase in the computational time. First, the definition of $q(\mathbf{u}_K)$ allows us to compute the KL in closed form, avoiding the need to resort to estimation by sampling and to compute the Jacobian of the transformation. Note that computing this Jacobian can be done in linear time for marginal flows although it will have, in general, a cubic cost (see Papamakarios et al. (2019) for a review on the key points of Normalizing Flows). Moreover, other choices of $q(\mathbf{u}_K)$ could require computation of the inverse $\mathbb{G}_\theta^{-1}(\mathbf{u}_K)$ to evaluate the density of the posterior sample under

$p(\mathbf{u}_0)\mathbf{J}_{\mathbf{u}_K}$. On the other hand, not canceling $p(\mathbf{f}_K|\mathbf{u}_K)$ would also require to approximate the determinant of the transformation $\mathbf{J}_{\mathbf{f}_K}$ and the inverse $\mathbb{G}_\theta$, with *the whole dataset* $\mathbf{X}$, something that cannot be done stochastically. So canceling this term is not only important to allow SVI, but also to avoid costly Jacobian/inverse computations.

Finally, the use of marginal flows and definition of the variational posterior allow us to analytically integrate out the inducing points – a very desirable property both for training and when making predictions.

B Additional Details on Experiments

B.1 Description of Flows

In this section we provide a description of the flows used throughout the experiments. We describe the 'base' transformations that are the building blocks to compositional and input-dependent flows, see Table. 1.

Table 1: Description of Flows

Flow	Forward	Inverse	Parameters
Log	$\log(\mathbf{f}_0)$	$\exp(\mathbf{f}_K)$	—
Exp	$\exp(\mathbf{f}_0)$	$\log(\mathbf{f}_K)$	—
Softplus	$\log(\exp(\mathbf{f}_0)+1)$	$\log(\exp(\mathbf{f}_K)-1)$	—
Sinh	$\sinh(\mathbf{f}_0)$	$\text{arcsinh}(\mathbf{f}_0)$	—
Arcsinh	$a\text{arcsinh}(b(\mathbf{f}_0+c))+d$	$\sinh(\mathbf{f}_K)$	$a,b,c,d\in\mathbb{R}$
Affine (L)	$a+b\cdot\mathbf{f}_0$	$\frac{\mathbf{f}_K-a}{b}$	$a,b:\mathbb{R}$
Sinh-Archsinh	$\sinh(b\cdot\text{arcsinh}(\mathbf{f}_0)-a)$	$\sinh(\frac{1}{b}(\text{arcsinh}(\mathbf{f}_0)+a))$	$a,b:\mathbb{R}$
Boxcox	$\frac{1}{\lambda}(\text{sgn}(\mathbf{f}_0)\mid\mathbf{f}_0\mid^{\lambda}-1)$	$\text{sgn}(\lambda\cdot\mathbf{f}_0+1)\mid\lambda\cdot\mathbf{f}_0+1\mid^{\frac{1}{\lambda}}$	$\lambda>0$
Tukey	$\frac{1}{g}[\exp(g\cdot\mathbf{f}_0)-1]\exp(\frac{h}{2}\mathbf{f}_0^2)$	—	$h\in R^+, g\in R: g\neq 0$
TanH	$a\tanh(b(\mathbf{f}_0+c))+d$	—	$a,b,c,d\in\mathbb{R}$

These flows may combined to construct compositional flows and their parameters may be input-dependent. Moreover, we can create a new flow by linear combination of any of these flows, where the parameters have to be restricted so that each individual flows are strictly increasing/decreasing functions. An example could be the following:

$$\mathbf{f}_K=\sum_i^I a_i+b_i\cdot\text{arcsinh}((\mathbf{f}_0-c_i)/d_i); b_i,d_i\in\mathbb{R}^+\ \forall i \tag{26}$$

Finally, as in Rios and Tobar (2019) we define SAL to be a Sinh-Archsinh flow with an Affine flow (L). We also consider compositions of flows made up directly by the inverse parametrization. For example in one of our experiments we experimented with the BoxCox+L flow and the InverseBoxCox+L flow. We provide this additional information in our GitHub.

B.2 Initializing Flows from Data

In this section we describe an initialization scheme that attempts to learn flow parameters that Gaussianize the prior. In the derivation we approximate the data as being noise-free and so in practice this may also be used for the likelihood transformations of the WGP. Ideally we would want to learn a normalizing flow $\mathbb{G}(\cdot)$ that transforms a standard Gaussian φ to the true prior $p(\mathbf{f}_0)$:

$$\varphi(\mathbb{G}^{-1}(\mathbf{f}))\frac{\partial\mathbb{G}^{-1}}{\partial\mathbf{f}}=p(\mathbf{f}_0) \tag{27}$$

In practice we do not have access to the true prior but instead observations $\mathbf{Y}$. By using $\mathbf{Y}$ as approximate samples from $p(\mathbf{f}_0)$ we can then optimize $\mathbb{G}$ to approximately satisfy Eq. 27. To optimize we directly minimize the KL divergence between $p(\mathbf{f}_0)$ and $\varphi(\mathbb{G}^{-1}(\mathbf{f}))\frac{\partial\mathbb{G}^{-1}}{\partial\mathbf{f}}$:

$$\text{KL}\left[p(\mathbf{f}_0)\mid\mid\varphi(\mathbb{G}^{-1}(\mathbf{f}))\frac{\partial\mathbb{G}^{-1}}{\partial\mathbf{f}}\right]=\mathbb{E}_{p(\mathbf{f}_0)}\left[\log\varphi(\mathbb{G}^{-1}(\mathbf{f}))\frac{\partial\mathbb{G}^{-1}}{\partial\mathbf{f}}\right]-\mathbb{E}_{p(\mathbf{f}_0)}\left[p(\mathbf{f}_0)\right] \tag{28}$$

The second term is constant w.r.t to the flow parameters and hence we only need to consider and optimize the first term. Because we have assumed that the $\mathbf{Y}$ are approximate samples from the true prior we write:

$$E_{p(\mathbf{f}_0)}\left[\log\varphi(\mathbb{G}^{-1}(\mathbf{f}))\frac{\partial\mathbb{G}^{-1}}{\partial\mathbf{f}}\right]\approx\sum_{n=1}^N\log\varphi(\mathbb{G}^{-1}(\mathbf{Y}_n))\frac{\partial\mathbb{G}^{-1}}{\partial\mathbf{Y}_n}=\sum_{n=1}^N\log\varphi(\mathbb{G}^{-1}(\mathbf{Y}_n))(\frac{\partial\mathbb{G}}{\partial\mathbb{G}(\mathbf{Y}_n)})^{-1} \tag{29}$$

and the final initialization optimization procedure is:

$$\arg\min_{\theta} \sum_{n=1}^{N} \log \varphi(\mathbb{G}^{-1}(\mathbf{Y}_n)) \frac{\partial \mathbb{G}^{-1}}{\partial \mathbf{Y}_n} \tag{30}$$

A similar derivation is used by Papamakarios et al. (2017) but in a different context.

B.3 Initializing Flows approximately to Identity

In this section we provide details on how we initialize flows close to identity. Many transformations can already recover identity but for those that cannot this method provides an effective and simple way to initialize them. To find these parameters we simply generate observations from the line $y = x : [x_n, y_n]_{n=1}^{N}$ and minimize the MSE loss of the flow mapping from x to y:

$$\arg\min_{\theta} \frac{1}{N} \sum_{n=1}^{N} (y_n - \mathbb{G}(x_n))^2 \tag{31}$$

B.4 Initialization of Input-dependent flows

To initialize input-dependent flows we first initialize standard (non-input-dependent) flow parameters $\hat{\theta}$ by any of the procedures described above. Then, we turn the parameters into input-dependent and initialize the NN parameters to match the values learned in the first step. This is done through stochastic gradient optimization, i.e. by first sampling a minibatch from the data distribution, and then minimize the empirical MSE loss between $\text{NN}(\mathbf{X})$ and $\hat{\theta}$.

B.5 Real World Experiments

For both real world experiments, we consider 2 different seeds, shuffle the observations and run 5-fold cross-validation across 2 different optimization schemes. The first optimization scheme optimizes both the variational and hyperparameters jointly. The second holds the likelihood noise fixed for 60% of iterations. This is to help avoid early local minimum that causes the models to underfit and explain the observations as noise.

For all models, we use RBF kernels with lengthscales initialized to 0.1, and Gaussian likelihoods with noise initialized to 1.0. We optimize the whitened variational objective using Adam optimizer with a learning rate of 0.01.

B.5.1 Air Quality

We used data from the London Air Quality Network London (2020) and we focus on site HP5 (Honor Oak Park , London) using 1 month of PM25 data (731 observations, date range 03/15/2019 - 04/15/2019). Because the observations are non-negative bounded we only consider the following positive enforcing flows: sal+softplus randomly initialized, sal+softplus initialized from data and a sal + sal + softplus initialized from data.

We shuffle observations and run 5-fold cross validation across 5%, 10% and 100% of inducing points and optimize each for a total of 10000 epochs. We compute means and standard deviations across all flows, folds and seeds. In the main paper we only present results using a sal+softplus (initialized from data) flow and we now present additional results across all these positive enforcing flows.

Additional results averaged across multiple flows

We now additionally present the results averaged across all the considered flows. The results echo the findings on the single flow experiment, that the TGP outperforms both the GP and V-WGP across the 3 levels of inducing points considered. This suggests that the results of the TGP are somewhat invariant to the choice of (positive enforcing) flows used.

B.5.2 Rainfall

The Switzerland rainfall uses data collected on the 8th of May 1986. Because all the observations are positively bounded we again use positive enforcing flows. We consider: softplus, sal+softplus (from data), sal+softplus

Figure 2: RMSE (left is better) on the Air quality dataset across 5%, 10% and 100% of inducing points.

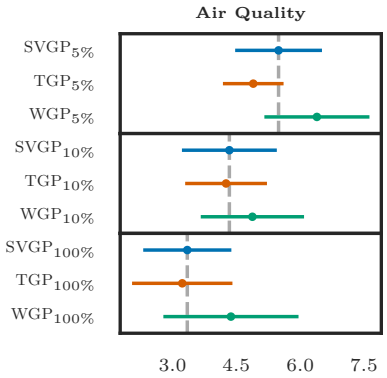


Table 2: Table of results to reproduce the left panel

Model	RMSE	Standard Deviation
GP 5	5.491	1.02
TGP 5	4.896	0.713
WGP 5	6.393	1.235
GP 10	4.335	1.115
TGP 10	4.257	0.961
WGP 10	4.876	1.214
GP 100	3.345	1.036
TGP 100	3.227	1.18
WGP 100	4.372	1.589

(from identity), sal+sal+softplus (from identity), sal+sal+softplus (from data).

We optimize for a total of 20000 epochs and compute means and standard deviations across all flows, folds and seeds.

Figure 3: RMSE (left is better) on the Rainfall experiment.

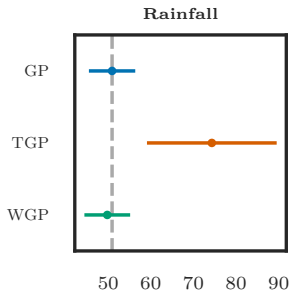


Table 3: Table of results to reproduce the left panel.

Model	RMSE	Standard Deviation
GP	50.892	5.441
TGP	74.306	15.225
WGP	49.764	5.369

B.5.3 Table of results in figures in main paper

In this section we present, for reproducibility and comparison reasons, the numerical values used to generate Fig. 6 in the main paper.

B.6 Black Box Results

In the black box experiments we explore the performance of TGP across many UCI datasets (Lichman, 2013). The performance measures are evaluated by employing random 10 fold train-test partitions and reporting average results plus standard error. This is done for all the datasets except Year and Avila (because the test partition is already provided) and Airline, where we just use a random 1 fold partition following previous works e.g (Salimbeni and Deisenroth, 2017). We perform flow selection by running each of the candidate models using random validation splits. We use one validation split on the first and second random fold partitions, except for Year and Airline where we only use one. This selection is done for 100 inducing points. The selected model is then used across all the experiments reported, including the experiments with less inducing points.

To initialize our models we follow Salimbeni and Deisenroth (2017). We use RBF kernels with parameters initialized to 2.0. The inducing points are initialized using the best of 10 KMeans runs except for Year and Airline where we just use 1 run. We use a whitened representation of inducing points and initialize the variational parameters to $\mathbf{m} = 0$ and $\mathbf{S} = 1^{-5}I$. The DGP have an additional white noise kernel added to the RBF in each hidden layer, with the noise parameter initialized to 1^{-6}. The noise parameter of the Gaussian likelihood is initialized to 0.05 for the regression experiments. For classification we use a noise free latent function and

Table 4: Table of results to generate Fig. 6 in Main paper. The left table shows results for the Air quality experiment (Left panel in Fig. 6) and the right table shows results for the rainfall experiment (Right panel in Fig. 6).

Model	RMSE	Standard Deviation
GP 5	5.491	1.02
TGP 5	4.851	0.786
WGP 5	6.451	1.207
GP 10	4.335	1.115
TGP 10	4.299	1.269
WGP 10	4.505	0.986
GP 100	3.345	1.036
TGP 100	2.833	0.923
WGP 100	4.151	1.702

Model	RMSE	Standard Deviation
GP	50.892	5.441
TGP SP	50.858	5.395
WGP SP	50.893	5.356
TGP SAL+SP	73.057	8.465
WGP SAL+SP	48.85	5.404

Bernoulli/Categorical likelihoods for Binary/Multiclass problems. We use probit and softmax link functions respectively. We use Adam optimizer with a learning rate of 0.01. The flow initializers are run over 2000 epochs for the identity initializer and over 2000 (rest-of-datasets) or 20 (Year-Airline) epochs for input-dependent flows, to match the learned parameters in the previous initialization step. In our experiments however, we observed that the flow could be initialized with fewer epochs. We have tried a set of different combinations of flows as described in Tab. 1. This includes different flow lengths and different number of flows in the linear combinations. For input-dependent flows we just tried the SAL flow with lengths 1 and 3; where dependency is encoded just in the non-linear flow (i.e the sinh-arcsinh). For these experiments we focus on exploring different architectures of the Neural Networks. We search over $\{1, 2\}$ hidden layers, $\{25, 50\}$ neurons per layer, $\{0.25, 0.5, 0.75\}$ dropout probabilities, batch normalization and ReLu and Tanh activations. We found that any of these possible combinations could work except the use of Batch Normalization, and that most of this combinations could be successfully optimized using the default optimizer, although some combinations suggested that a lower learning rate was needed to make optimization stable (this combinations were discarded as we wanted to show robustness against optimizer hyperparameter search). The prior over the neural network parameters is kept fixed and is introduced in the model by fixing a 1^{-5} weight decay in the optimizer ($\lambda = 1^{-5}$). In our Github we provide additional information about the model selection process and the final selected models. All of our models were optimized for 15000 epochs for all datasets except Year and Airline where we use 200 epochs. Each epoch corresponds to a full pass over the dataset. For classification we follow Hensman et al. (2015) and freeze the covariance parameters before learning everything end to end. This is done for the first 2000 epochs.

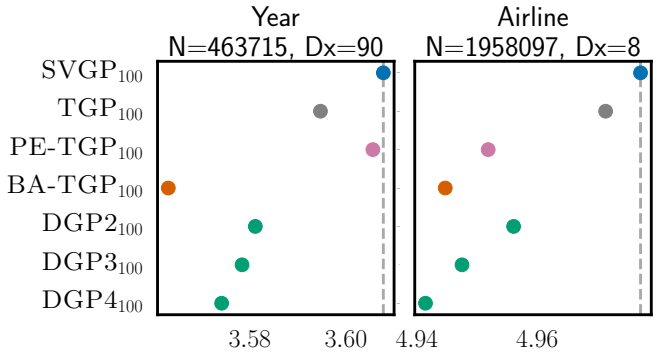
For experiments using less than 100 inducing points we use the same flow architecture selected from the validation sets and optimizer hyperparameters as the corresponding 100 inducing point experiment. The performance obtained highlights that our approach is somewhat robust to hyperparameter selection. On just one dataset we observe that this extrapolation was suboptimal and that the algorithm diverge. Those results are not reported in this appendix and correspond to 5 inducing points, input-dependent flows and `naval` dataset. We attribute this fact to not having performed model selection and optimizer hyperparameter search for each set of inducing points specifically. On the other hand, if in any of the experiments carried out failed due to numerical errors (e.g Cholesky decomposition) we increase the standard amount of jitter added by Gpytorch from 1^{-8} to 1^{-6}. This is just needed on some train-test folds and some datasets only when using less inducing points. In general we found that our experiments were quite stable.

B.6.1 Regression

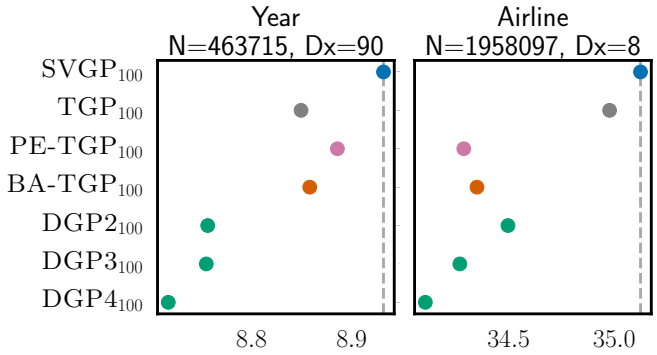
In this subsection we present the complete results across the regression benchmarks. This includes the NLL and RMSE for different numbers of inducing points on the following models: Sparse Variational GP (SVGP), Transformed GP with non input-dependent flows (TGP), Transformed GP with point estimate input-dependent flows (PE-TGP) and Transformed GP with Bayesian input-dependent flows (BA-TGP). Note that PE-TGP and BA-TGP share the same input-dependent architecture and learned parameters $\mathbf{W}$, and the only difference relies on whether we use standard Dropout (Srivastava et al., 2014) or Monte Carlo dropout (Gal and Ghahramani, 2016) to make

predictions.

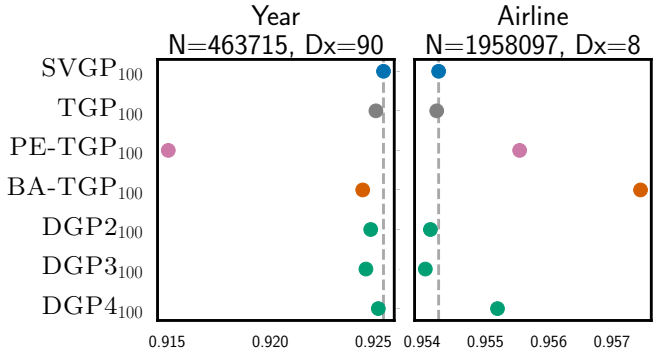
Large Scale Regression We report the NLL and RMSE for the Large scale regression problem in Fig. 4a and Fig. 4b. Across both datasets our model outperforms the baseline GP both in RMSE and NLL. Furthermore, the TGP also performs well, although in general is outperformed by the BA-TGP. These plots also show the regularization effect of Bayesian marginalization. In Year the RMSE and NLL is highly improved when accounting for uncertainty in the parameters. In Airline both models provide similar RMSE, but the BA-TGP provides better uncertainty quantification, reflected by improved NLL scores. In these experiments the DGP mostly outperforms TGP, PE-TGP and BA-TGP except for on the Year dataset where BA-TGP has a lower NLL, indicating better uncertainty quantification.



(a) NLL (left is better) for large scale regression datasets.



(b) RMSE (left is better) for large scale regression datasets.



(c) 95% COVERAGE (right is better) for large scale regression datasets.

Medium-Small Regression We now illustrate the results for the medium small regression in Fig. 5 (NLL) and Fig. 6 (RMSE). We show results split across decreasing number of inducing points and different models. Across all levels of inducing points, the BA-TGP model ranks the best and consistently outperforms alternative models on both NLL and RMSE showing superior point-prediction and uncertainty quantification.

On the other hand, by looking at e.g `power` dataset, and comparing RMSE and NLL, we can see how in terms of

RMSE both the PE-TGP and BA-TGP perform similarly. However, there is a big difference in terms of NLL, which is an indicator of good predictive uncertainty quantification provided by introducing uncertainty in the flow parameters. We build on this observation in the final subsection of this appendix.

Moreover, a particularly interesting outcome is the performance of the models when only using 5 inducing points. We can see that in `kin8nm`, `power` and `concrete` the 5 inducing points provides a similar performance to the 100 inducing points for the BA-TGP. We show in the subsequent section that even though the Neural network is highly expressive, the base GP is necessary and not 'ignored' by the model. Hence we can attribute the excellent performance of BA-TGP to both the combination of the BNN and the GP.

We can also see how the standard TGP is also able to improve upon the GP in some datasets, although the improvement is minimal, clearly highlighting the necessity of input-dependent flows. The fact that the standard TGP has been tested using more complex transformations than the input-dependent TGP (which uses just 1-3 length SAL flows) suggest that the boost in performance clearly comes from the input dependency and not the warping function. This means that these results can be improved by testing more complex input-dependent flow combinations, which is something we let for future work. Also we can see in `wine-red` that while the uncertainty quantification of our model and the DGP is quite similar, we clearly outperform it in terms of pointwise predictions. The DGP consistently outperforms the TGP and SVGP w.r.t RMSE but in general the BA-TGP and PE-TGP achieve superior performance. These two observations might indicate that the proposed model is more expressive in terms of pointwise predictions than a DGP.

Finally, note how without doing specific model selection for the less inducing points models, the parameters extrapolated from the 100 inducing points one works very well, which means that our model is somewhat robust to the selection of hyperparameter. Also by noting that all the models are trained for 15000 epochs, we show how our model has not over-fit, although being much more complex than a 'simple' GP.

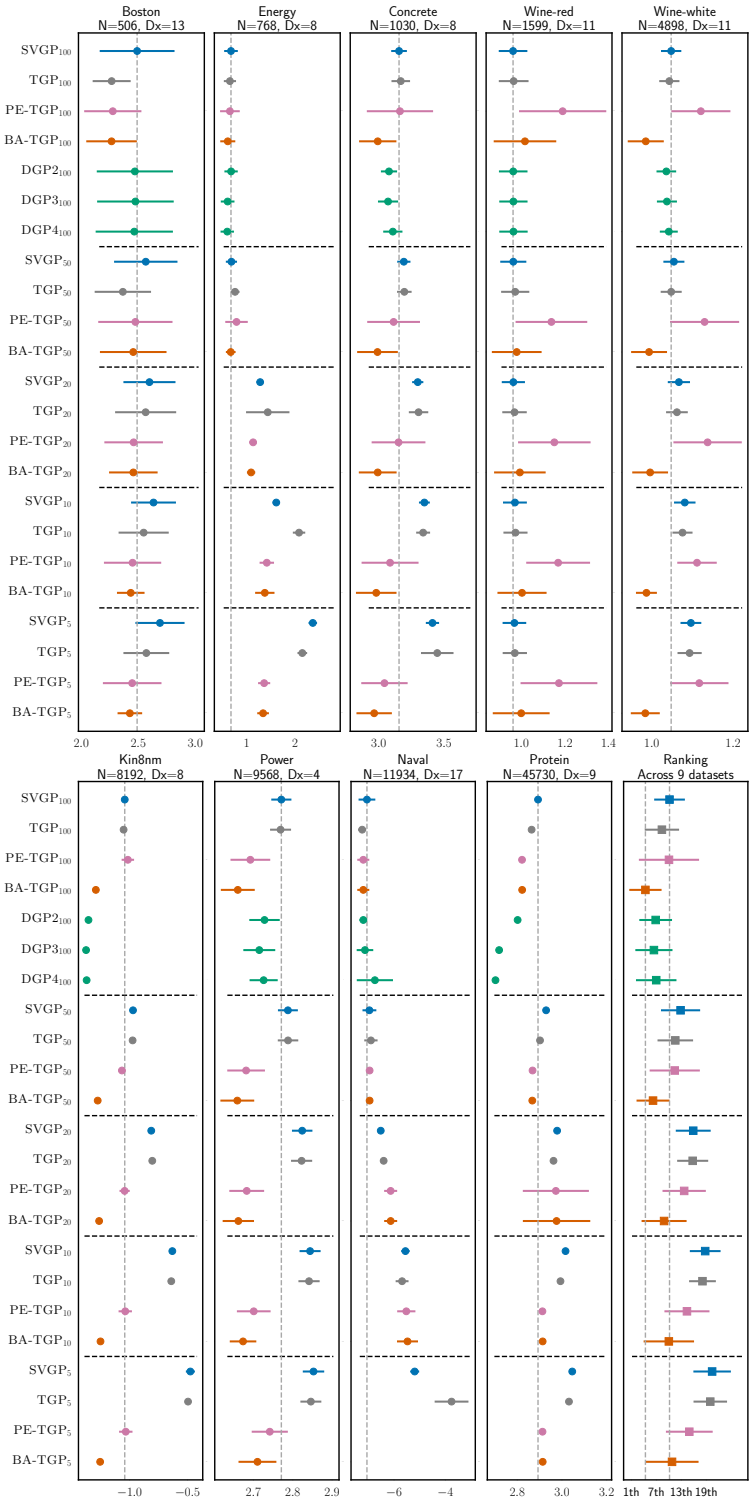


Figure 5: Comparing NLL (left is better) across 9 data sets for several number of inducing points. **Bottom right panel:** Ranking of the methods across all 9 data sets. TGP stands for non input-dependent flows, PE-TGP stands for point estimate input-dependent flows (Standard Dropout) and BA-TGP stands for the Bayesian input-dependent flows (MC Dropout)

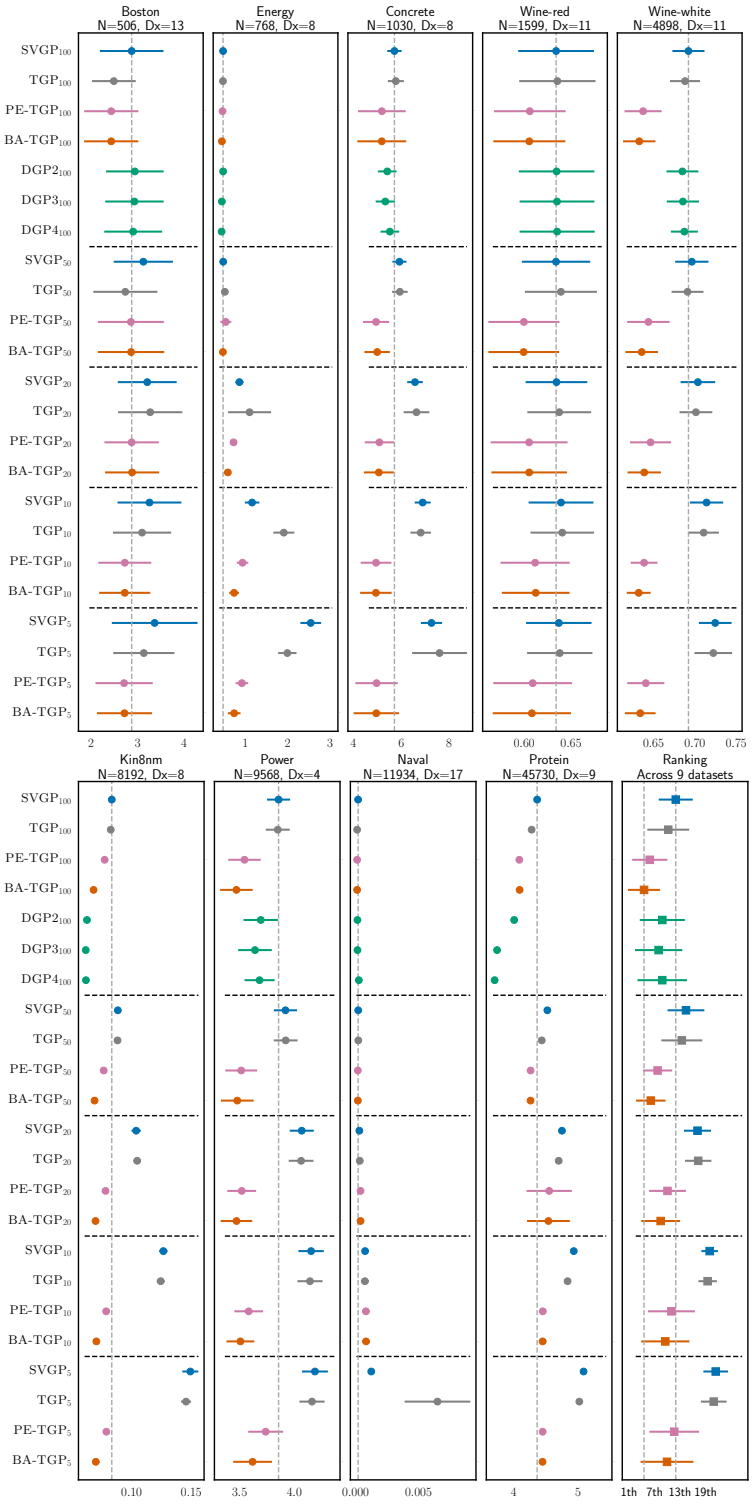


Figure 6: Comparing RMSE (left is better) across 9 data sets for several number of inducing points. **Bottom right panel:** Ranking of the methods across all 9 data sets. TGP stands for non input-dependent flows, PE-TGP stands for point estimate input-dependent flows (Standard Dropout) and BA-TGP stands for the Bayesian input-dependent flows (MC Dropout).

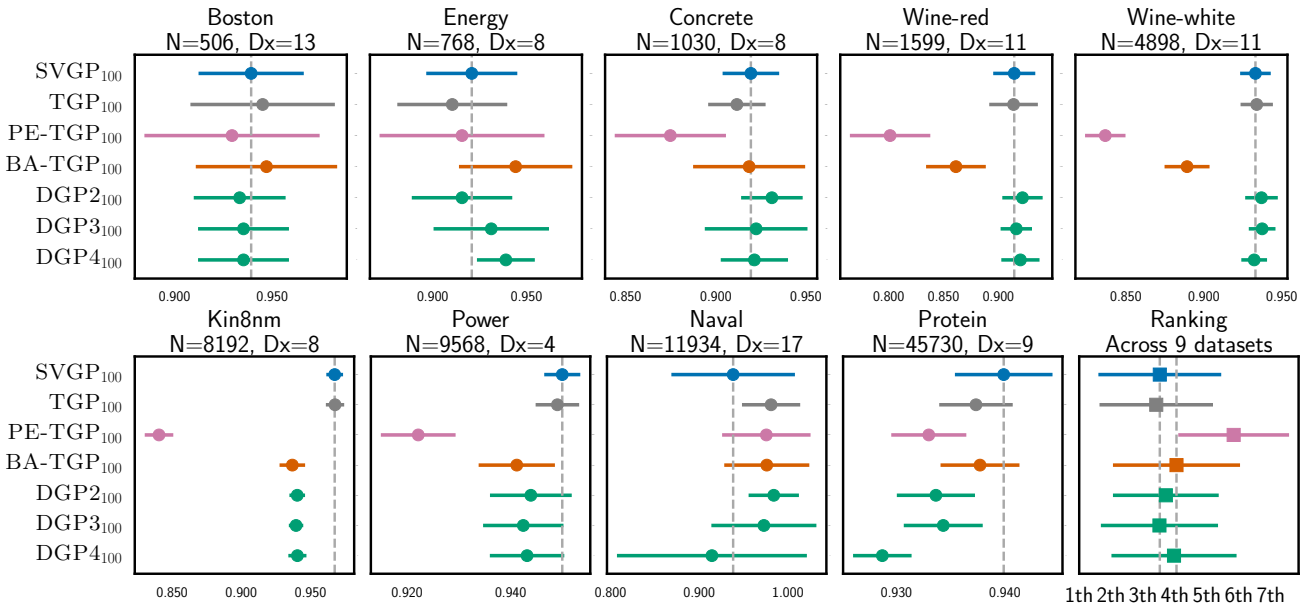


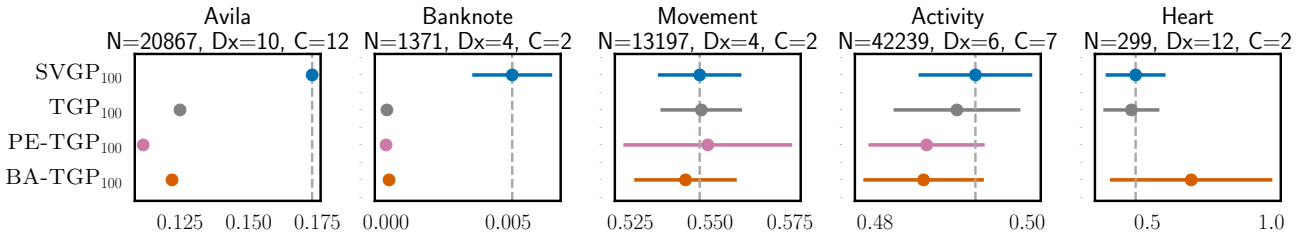
Figure 7: Comparing 95% COVERAGE (right is better) across 9 data sets for 100 inducing points. **Bottom right panel:** Ranking of the methods across all 9 data sets. TGP stands for non input-dependent flows, PE-TGP stands for point estimate input-dependent flows (Standard Dropout) and BA-TGP stands for the Bayesian input-dependent flows (MC Dropout).

B.6.2 Classification

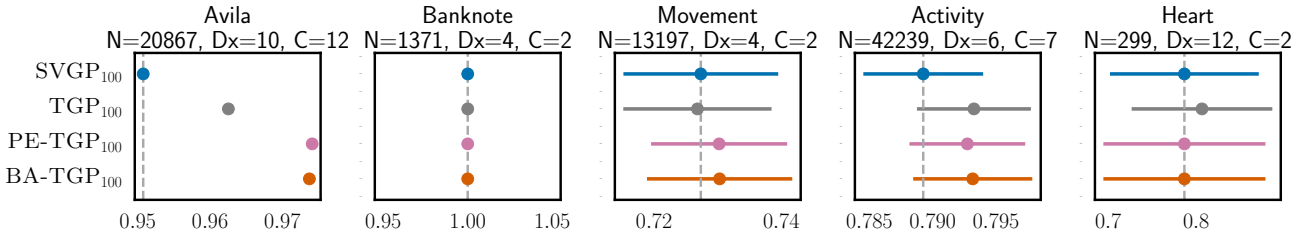
In this section we expand on the classification experiments from the main paper by providing two additional datasets and reporting accuracy. For all the datasets we show NLL in Fig. 8a and accuracy in Fig. 8b. Across all datasets our proposed models (either through input-dependent or non-input-dependent flows) outperform the SVGP, and only in `heart` does the TGP substantially outperform the input-dependent models BA-TGP and PE-TGP. We highlight the big boost in accuracy provided by our models (see e.g `avila` dataset). Surprisingly we observe that the PE-TGP performs well in classification and usually outperforms the BA-TGP. However, we have observed that sometimes the PE-TGP outputs extreme wrong values, giving a NLL of ∞. For the same model we observe that the BA-TGP was able to remove those extreme predictions. We speculate that the model is correctly incorporating epistemic uncertainty relaxing extremely wrong assigned confidences. On the other hand, we attribute the bad performance of the BA-TGP and PE-TGP in the `heart` dataset to prior misspecification. First note that L in this case theNN is not depicted for the PE-TGP, as this is one of the cases in which many predictions were extremely wrong, yielding a ∞ NLL. We can see how the BA-TGP solves this, but due to this misspecification is unable to provide good predictions. As explained by Knoblauch et al. (2019), prior misspecification leads to a misleading quantification of uncertainty. Further, the number of training data points is small, meaning that the prior has a relatively strong influence relative to the likelihood terms. In situations like this, a badly specified prior dominates the likelihood terms and adversely affects the predictive likelihoods.

To build on this claim we note that the TGP outperforms the GP on this dataset indicating that having a more expressive prior is beneficial. This coupled with the fact that we did not tune prior $p(\mathbf{W})$ for our BNN, and that this is the only dataset in which the BA-TGP does not give a clear boost in performance, are consistent observations with the hypothesis of prior misspecification.

Finally on `banknote`, we see that all the models provide a 100% accuracy, which means that the improvement in the NLL is coming from reducing the calibration gap, as the NLL is a proper scoring rule. Our model is making the predictions extreme towards the correct class, which is a desirable property if your data distribution doesn't present overlap between classes. Note however how in this case the BA-TGP still provides uncertainty in the predictions, avoiding the extreme $\{0, 1\}$ probability assignments.



(a) NLL (left is better) of all the classification datasets



(b) Accuracy (right is better) of all the classification datasets

Figure 8: Results for classification datasets. TGP stands for non input-dependent flows, PE-TGP stands for point estimate input-dependent flows (Standard Dropout) and BA-TGP stands for the Bayesian input-dependent flows (MC Dropout).

B.6.3 Inference with Variational Bayes

We run a subset of the experiments using variational bayes inference instead of Monte Carlo dropout, using a standard normal prior $p(\mathbf{W}) = \prod_{\mathbf{w}_i}^{|\mathbf{W}|} \mathcal{N}(\mathbf{w}_i|0, 1)$ over the weights of the Neural Network and the same Neural

Network architecture used by the MC dropout experiments to parameterize the input dependency (with the exception of the dropout layer that is removed).

We initialize the flows following the proceedure described in Sec. B.4: The variational parameters from $q(\mathbf{W})$ are optimized to minimize the sum of squared errors between the output of the neural network and the value of the parameter that makes the flow learn an identity mapping, i.e. we do not incorporate the KL penalty from the ELBO at this point. We found that this initialization procedure is harder to optimize as opposed to Monte Carlo dropout, and we let this for future work, as this could depend not only on the inference method but on the kind of flow we use (1-3 length SAL flows).

As a result of the initialization most of the models did not achieve a good starting point for the subsequent optimization and this makes the model either provide suboptimal results or directly make optimization unstable. We run experiments on the following datasets using just 5 seeds: `boston`, `energy`, `concrete`, `kin8nm` and `wine_red`. We found that in all of them but `boston` and `concrete` the model saturated, and that in `concrete` the optimization was very unstable (even if we lower the learning rate), which leads to bad local optima. As shown in table below this made the model provide unseful predictions (around $16.x$ values of RMSE).

Beyond the initialization procedure the reasons behind these problems can be attributed to only using one Monte Carlo sample to estimate the ELBO during training. On the other hand, using more Monte Carlo samples for training implies higher training time, while we are interested in efficient implementations of our model that are much faster than a DGP. Furthermore, the results showed in the next table suggest that beyond this computational and optimization issues, MC dropout seems to provide better results than Variational Bayes under the same training specifications (1 sample for training, same likelihood parameterization etc), hence the study of possible alternatives for inference is something we leave for future work.

		SVGP	Point Estimate TGP	MC dropout TGP	Variational Bayes TGP
Concrete	NLL	3.17	3.24	3.02	5.67
	RMSE	5.79	5.55	5.57	18.02
Boston	NLL	2.38	2.26	2.24	2.39
	RMSE	2.66	2.38	2.33	2.60

B.7 Bayesian Flows

In this subsection we provide additional information about Bayesian flows to highlight and connect with the conclusions provided in the UCI datasets section. We first show the effect of considering parameter uncertainty in the flow. We then illustrate why the whole modeling is not done by the BNN.

B.7.1 Uncertainty in the Warping Function

To illustrate the parameter uncertainty introduced by using a BNN we plot the warping function evaluated at four different training locations $\mathbf{X}^{(n)}$ for the `power` dataset. This means that we will show four different warping functions $\mathbb{G}$ with parameters given by $\{\theta(\mathbf{W}, \mathbf{X}^{(i)})\}_{i=1}^{4}$. Note that the weights of the $\mathbf{W}$ are shared and the difference in each of the parameters comes from the specifics inputs $\mathbf{X}$. The figures show the function parameterized by the different warping functions when applied on $\mathbf{f}_0$, i.e. we plot $\mathbf{f}_K = \mathbb{G}_{\theta(\mathbf{W}, \mathbf{X}^{(n)})}(\mathbf{f}_0)$. The range $\{\mathbf{f}_{0_a}, \mathbf{f}_{0_b}\}$ in which the flow is evaluated goes from the minimum $\mathbf{Y}$ value in the training dataset to the maximum one. In this way we show what kind of warping function has the model learned for the output range to be regressed.

The top row in Fig. 9 shows the transformation $\mathbb{G}_{\theta(\mathbf{X}^{(n)}, \mathbf{W})}$ for the point estimate flow. The bottom row shows the mean and the standard deviation of the same flow where the parameters have been sampled from the posterior distribution. Note that the top and bottom rows use the same Neural Network parameters. The only difference is that while in the top row we compute the flow parameters with one forward pass through the Neural Network, by multiplying the activations by the 1 minus the probability of dropout p (Srivastava et al., 2014), in the bottom row we drop activations with probability p on each forward through the Neural Network (Gal and Ghahramani, 2016).

First, we can see how the model learns a different warping function for each different input $\mathbf{X}^{(n)}$, and this means that the marginal distribution at each index of our stochastic process is different. Second, we observe how the

Bayesian flow incorporates parameter uncertainty. Note that both the Bayesian and point estimate flows have the same mean function; and this is the reason why the RMSE metrics reported in Fig. 6 for the `power` dataset are very similar (note that RMSE only considers the first moment of the posterior predictive). However the NLL showed in Fig. 5 is much better for the Bayesian flow, as we are incorporating epistemic uncertainty, hence being less confident in regions of function evaluations $\mathbf{f}_0$ where the model is uncertain. Third, note that the functions plotted for each of the training samples are simple, because the functional form is just given by a $K = 3$ input-dependent flow. Hence input-dependent flows are not overfitting because the functional transformation $\mathbb{G}$ is complex, but because it is very specific for a particular point $\mathbf{X}^{(n)}$. By incorporating parameter uncertainty in the flow parameters we are considering all the possible functions parameterized by the flow's functional $\mathbb{G}$, hence preventing overfitting. We can expect this regularization effect to be bigger when making the functional's form more complex, which is something we let for future work.

B.7.2 Uncertainty handled by the GP

In the UCI experiment section we comment that sometimes the model with 5 inducing points provides similar results to the 100 inducing points models. Initially we could think that the BNN is handling all the modeling power both in terms of uncertainty quantification and regressed values. In this subsection we illustrate that this is not the case and that the modeling performance comes from a combination of the NN and the GP. Note that

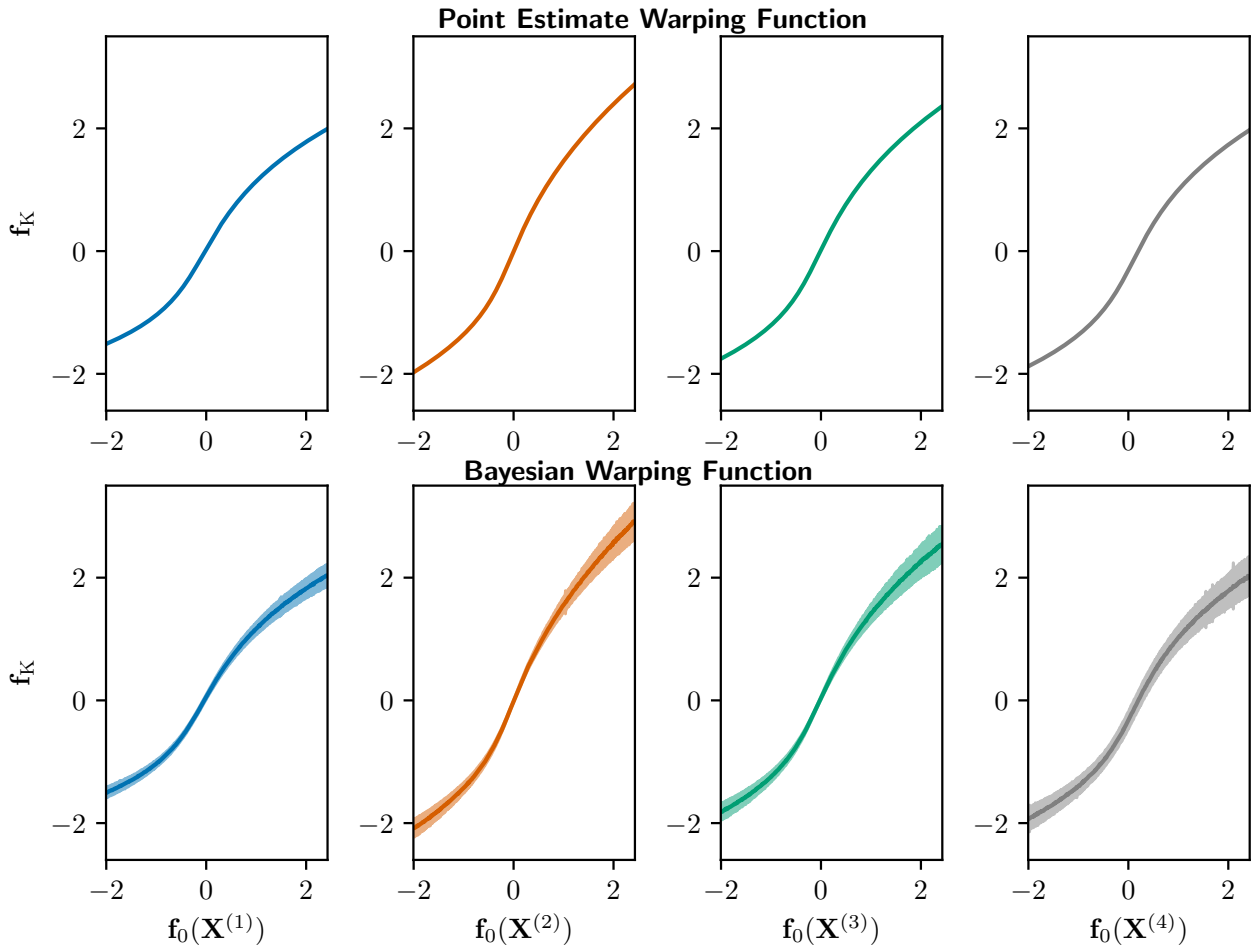
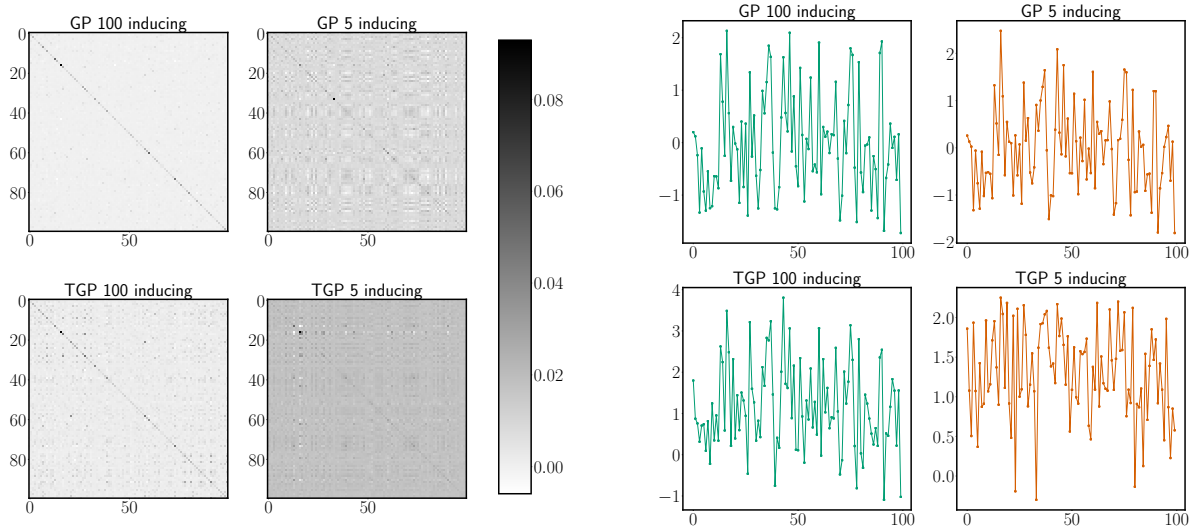


Figure 9: Example of warping functions obtained with input-dependent flows for the `power` dataset. The top row shows the point estimate warping function evaluated over a range $\mathbf{f}_0$, at different input locations using standard Dropout. The bottom row shows mean and standard deviation of samples from the posterior of the Bayesian flow using Monte Carlo Dropout. We can see how the model learns a different function warping depending on the input locations and how the model accounts for parameter uncertainty.



(a) Covariance from $q(\mathbf{f}_0)$ evaluated at 100 training points

(b) Mean from $q(\mathbf{f}_0)$ evaluated at 100 training points

Figure 10: This figure shows the mean and covariance from the GP variational distribution $q(\mathbf{f}_0)$ evaluated at 100 training points. As shown in the plot the covariance have not collapse to a point mass (i.e the cells would have to be completely white) and the mean also change across training points. This means that the model has not learned to just output a constant value for $\mathbf{f}_0$ and model everything through the Neural Network.

the uncertainty provided by the GP is combined with the uncertainty provided by the BNN.

To do so, we pick the `concrete` dataset, which is one of the datasets in which this effect is presented. For this dataset we plot the mean and covariance of $q(\mathbf{f}_0)$ at 100 random training locations in Fig. 10. As we see in the plot, the mean and covariance from $q(\mathbf{f}_0)$ has not collapsed to a constant distribution. This means that the model has not learn to output a constant value for $\mathbf{f}_0$ and perform all the modelling through the input-dependent $\theta(\mathbf{W}, \mathbf{X}^{(n)})$ model. Note however that understanding the specific role of the GP and the BNN in our model in terms of uncertainty quantification is something that we leave for future work.

References

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

Gal, Y. and Ghahramani, Z. (2016). 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*, ICML'16, page 1050–1059. JMLR.org.

Hensman, J., de G. Matthews, A. G., and Ghahramani, Z. (2015). Scalable variational gaussian process classification. In Lebanon, G. and Vishwanathan, S. V. N., editors, *AISTATS*, volume 38 of *JMLR Workshop and Conference Proceedings*. JMLR.org.

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

Kingma, D. P. and Welling, M. (2014). Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Knoblauch, J., Jewson, J., and Damoulas, T. (2019). Generalized variational inference: Three arguments for deriving new posteriors. *arXiv preprint arXiv:1904.02063*.

Lichman, M. (2013). UCI machine learning repository.

London, I. C. (2020). Londonair - london air quality network (laqn). `https://www.londonair.org.uk`.

Opper, M. and Archambeau, C. (2009). The variational gaussian approximation revisited. *Neural Comput.*, 21(3):786–792.

Papamakarios, G., Nalisnick, E. T., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2019). Normalizing flows for probabilistic modeling and inference. abs/1912.02762.

Papamakarios, G., Pavlakou, T., and Murray, I. (2017). Masked autoregressive flow for density estimation. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 2338–2347. Curran Associates, Inc.

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

Rios, G. (2020). Transport gaussian processes for regression.

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

Salimbeni, H. and Deisenroth, M. (2017). 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., editors, *Advances in Neural Information Processing Systems 30*, pages 4588–4599. Curran Associates, Inc.

Snelson, E., Rasmussen, C. E., and Ghahramani, Z. (2003). Warped gaussian processes. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, page 337–344, Cambridge, MA, USA. MIT Press.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Titsias, M. (2009). Variational learning of inducing variables in sparse gaussian processes. volume 5 of *Proceedings of Machine Learning Research*, pages 567–574, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA. PMLR.