Latent Derivative Bayesian Last Layer Networks

Joe Watson*[†] **Jihao Andreas Lin*[†]** **Pascal Klink[†]** **Joni Pajarinen[†‡]** **Jan Peters[†]**

[†] Department of Computer Science, Technical University Darmstadt

[‡] Department of Electrical Engineering and Automation, Aalto University

Abstract

Bayesian neural networks (BNN) are powerful parametric models for nonlinear regression with uncertainty quantification. However, the approximate inference techniques for weight space priors suffer from several drawbacks. The 'Bayesian last layer' (BLL) is an alternative BNN approach that learns the feature space for an exact Bayesian linear model with explicit predictive distributions. However, its predictions outside of the data distribution (OOD) are typically overconfident, as the marginal likelihood objective results in a learned feature space that overfits to the data. We overcome this weakness by introducing a functional prior on the model's derivatives w.r.t. the inputs. Treating these Jacobians as latent variables, we incorporate the prior into the objective to influence the smoothness and diversity of the features, which enables greater predictive uncertainty. For the BLL, the Jacobians can be computed directly using forward mode automatic differentiation, and the distribution over Jacobians may be obtained in closed-form. We demonstrate this method enhances the BLL to Gaussian process-like performance on tasks where calibrated uncertainty is critical: OOD regression, Bayesian optimization and active learning, which include high-dimensional real-world datasets.

1 Introduction

Bayesian neural networks (BNN) [43, 50] offer the possibility of combining the expressivity of neural networks with the principled uncertainty quantification and reg-

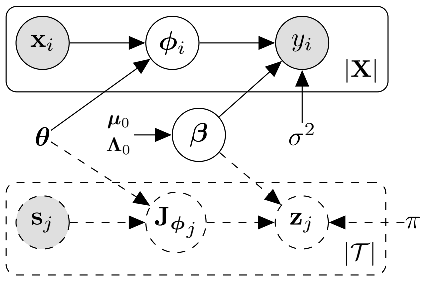

Figure 1: Graphical model of the Gaussian latent derivative Bayesian last layer network. Bottom dashed components indicate the latent derivative extension to the BLL (top).

ularization derived from Bayesian methods. However, inference for priors over the weights is intractable, resulting in extensive study of learning such BNNs via approximate inference [50, 30, 19, 25, 38, 7]. Despite their many varieties, these approximate models can suffer from several drawbacks, such as unintuitive priors, expensive training procedures, inaccurate posteriors and/or large model parameter spaces. Moreover, these models are typically restricted to sampling from implicit predictive densities, and have been criticized for their inaccurate uncertainty quantification [24, 23, 53, 54, 81, 85]. In many risk-sensitive and safety-critical applications, such as in medical diagnosis [21] and model-based control [17], well-calibrated predictive uncertainty is essential when the model is used outside of the data distribution (OOD). In contrast to Bayesian neural networks, Gaussian processes (GP) offer exact nonlinear, non-parametric Bayesian modeling through linear regression of a rich (often infinite) feature space, specified by a derived kernel function [63]. While a powerful and popular tool for probabilistic modeling [29], exact inference computation does not scale gracefully for large datasets, and the model's quality depends heavily on the choice of kernel given the data. Moreover, some kernels have been shown to suffer from the curse of dimensionality due to their use of distance metrics in the data space [4]. Despite sparse methods improving scalability [78, 72], parametric models still provide an attractive offer of

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130. Copyright 2021 by the author(s).

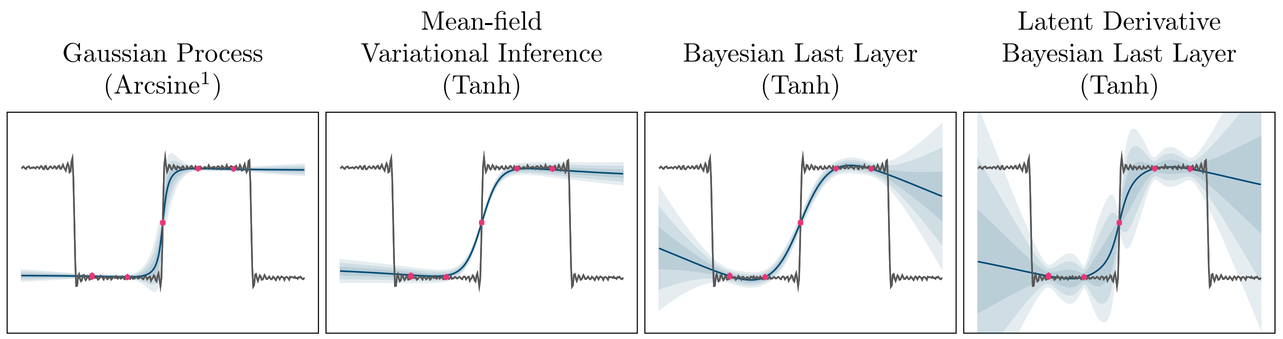

Figure 2: A toy example depicting Bayesian modeling of a complex function (——) from sparse data (•). The Gaussian process has well-calibrated uncertainty, but its fixed kernel structure can result in an undesirable function space for inference. Weight space mean-field variational inference (VI) exhibits poor in-between uncertainty [24, 23] and has an implicit predictive density. The Bayesian last layer has an *explicit* density and a deterministic feature space. However, these features overfit, resulting in poor epistemic uncertainty too. We introduce a 'latent derivative' prior to the BLL that encourages variance in the model's Jacobian distribution, diversifying the feature space by capitalizing on the network's large hypothesis space. This diversity results in increased predictive uncertainty, without sacrificing the fit.

flexible model specification with data-independent computation and parameterization.

The Bayesian last layer [39, 51] is an alternative BNN approach in the spirit of GPs, combining a Bayesian linear model with a learned, finite feature space, represented by a neural network. While the linear model ensures the analytical tractability of both inference and predictive distribution, the neural features provide the broad, adaptive hypothesis space offered by neural network architectures. Since Gaussian processes are just Bayesian linear models in an expressive feature space, can't learned neural features perform as well as kernels? While the network can be trained easily using gradient descent on the negative marginal likelihood, i.e. type-II maximum likelihood, there is a catch: The overparameterization leads to overfitting of the feature space [39, 63], resulting in a reduced hypothesis space of functions which severely limits the predictive uncertainty quantification (Figure 2).

To encourage diversity in the BLL's neural features, without sacrificing the model's attractive properties, we incorporate a novel *functional* prior into the model specification. We posit that well-calibrated uncertainty quantification may be effectively characterized by the distribution of the model's Jacobian w.r.t. the network's inputs (Figure 3), which is also a Gaussian process [64, 75]. Previous methods to improve BNN uncertainty quantification rely on additionally modeling the data distribution, requiring OOD samples to explicitly boost the predictive uncertainty [27]. The derivative prior works in- and outside the data distribution, influencing the model's hypothesis space and therefore *epistemic* uncertainty directly. By incorporating this prior into the objective, using the functional KL divergence (fKL) [68, 16, 76], the smoothness and diversity of the feature space is influenced by the variance of the prior. As a result, this training procedure resembles functional variational inference (fVI)[76]. Due to the BLL's deterministic features, the Jacobian may be computed directly using forward mode automatic differentiation (AD) [61], and the distribution over Jacobians can be obtained in closed-form thanks to the Bayesian last layer. However, during training the divergence to the functional prior must be approximated using samples. Moreover we believe this prior is intuitive, and its functional nature should enable the model to remain suitably calibrated independent of model size, as Bayesian models should [62].

By combining the analytic convenience of the Bayesian last layer, forward mode automatic differentiation and the novel functional prior over the Jacobian, we present a practical, calibrated Bayesian neural network that offers comparable utility to Gaussian processes, across small and large tasks. This class of BNN makes the case, like GPs, that priors are *not* required over the feature parameters. This reduction of complexity is motivated for applied domains such as robotics, where fast, well-calibrated Bayesian models are needed for sample-efficient, safe and risk-averse settings such as model-based reinforcement learning [18]. In these domains, the balance between simplicity and performance is key for practical use, and ideally not dependent on the amount of data in the task. To evaluate the benefit of this prior, we compare against standard BLLs and other baselines for OOD regression, active learning and Bayesian optimization, where epistemic uncertainty

[1]Note that the arcsine kernel is equivalent to an infinite hidden layer network with erf activations, therefore shares a similar hypothesis space to sigmoid and tanh networks.

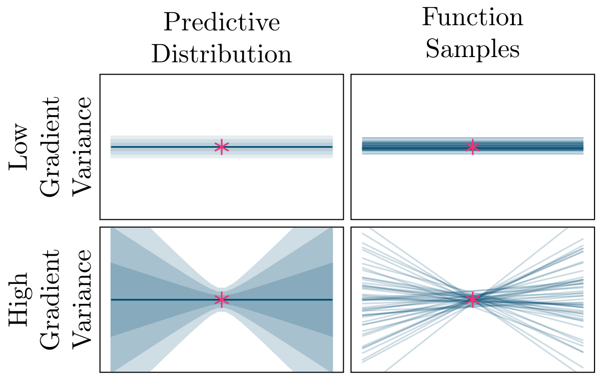

Figure 3: At a data point $*$, the epistemic uncertainty of a (locally) linear model is increased about the point if the variance of the function gradients is large.

is needed. We show that the latent derivative (LD) prior enhances the BLL's predictive uncertainty, scaling to large, real-world datasets with high-dimensional inputs, achieving superior or comparable performance on key tasks. We also introduce two new benchmarks for OOD prediction and active learning that utilize existing datasets from real-world robotic systems.

2 The Bayesian Last Layer

Intuitively, Bayesian last layer networks can be viewed as Bayesian linear regression in a projected feature space, where the projection is learned by a neural network. Alternatively, they can be thought of as a neural network whose parameters of the last layer are integrated out via exact, analytical Bayesian inference. While the BLL can be easily deployed for multivariate regression, the following derivations (and later experiments) in this work focus on univariate targets for simplicity.

Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ be the observed data, $\mathbf{x}_i \in \mathbb{R}^k$, $y_i \in \mathbb{R}$, such that $\mathbf{X} \in \mathbb{R}^{n\times k}$ and $\mathbf{y} \in \mathbb{R}^n$. Additionally, let $\boldsymbol{\phi}(\cdot;\boldsymbol{\theta})\colon \mathbb{R}^k \to \mathbb{R}^m$ be a feature space projection with parameters $\boldsymbol{\theta}$, $\boldsymbol{\phi}_i = \boldsymbol{\phi}(\mathbf{x}_i;\boldsymbol{\theta})$ and $\boldsymbol{\Phi} = \left[\boldsymbol{\phi}_1^\top \dots \boldsymbol{\phi}_n^\top\right] \in \mathbb{R}^{n\times m}$, the matrix of vertically stacked row vectors. It is common to add constant or linear terms to $\boldsymbol{\Phi}$ to implicitly represent bias or identity terms. However, for notational clarity, we ignore these terms and denote the projected feature space as $\mathbb{R}^m$.

A latent function f is modeled using Bayesian linear regression [9, 6, 49] with weights $\boldsymbol{\beta}$ and additive, zero-mean, Gaussian noise ϵ with variance σ^2, where

$$y_i = f(\mathbf{x}_i;\boldsymbol{\theta}) = \boldsymbol{\phi}_i^\top \boldsymbol{\beta} + \epsilon_i. \tag{1}$$

Placing a conjugate Gaussian prior $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Lambda}_0^{-1})$ over $\boldsymbol{\beta}$ results in a Gaussian posterior $\mathcal{N}(\boldsymbol{\mu}_n, \boldsymbol{\Lambda}_n^{-1})$ with an explicit Gaussian predictive distribution for query $\mathbf{x}$ [6],

$$y \mid \mathbf{x}, \mathcal{D}, \boldsymbol{\theta} \sim \mathcal{N}(\cdot \mid \boldsymbol{\phi}_\mathbf{x}^\top \boldsymbol{\mu}_n, \sigma^2 + \boldsymbol{\phi}_\mathbf{x}^\top \boldsymbol{\Lambda}_n^{-1} \boldsymbol{\phi}_\mathbf{x}), \tag{2}$$

where $\boldsymbol{\mu}_n$ and $\boldsymbol{\Lambda}_n$ are the mean vector and precision matrix of the posterior weight distribution.

The observation noise σ^2, prior weight parameters $\boldsymbol{\mu}_0$ and $\boldsymbol{\Lambda}_0$, and $\boldsymbol{\theta}$ can be either set to constants or optimized jointly by maximizing the log-marginal likelihood. With $\boldsymbol{\mu}_0 = \mathbf{0}$, this model is equivalent to a Gaussian process with kernel $k(\mathbf{x},\mathbf{x}';\boldsymbol{\theta}) = \boldsymbol{\phi}(\mathbf{x};\boldsymbol{\theta})^\top \boldsymbol{\Lambda}_0^{-1} \boldsymbol{\phi}(\mathbf{x}';\boldsymbol{\theta})$ [63]. For a more Bayesian treatment, an inverse gamma prior can be placed on σ^2, inducing a Student-t weight posterior and predictive density (Section A). We use 'GBLL' and 'TBLL' to differentiate these two approaches.

3 Latent Derivative Priors

For the Bayesian last layer, the neural features $\boldsymbol{\phi}$ are optimized using type-II maximum likelihood on the marginal likelihood (Equation (18)). While type-II ML can be effective at tuning hyperparameters, e.g. the lengthscale of a kernel, optimizing too many parameters runs the risk of overfitting [63]. For the BLL, this manifests as the feature space converging about the mean function. While this results in adequate uncertainty far away from the data, predictions are overconfident between datapoints (Figure 2).

To improve the diversity of the feature space, we are motivated to augment the marginal likelihood objective to leverage the expressiveness of the neural network without sacrificing fit. In this work, we build on the intuition that the distribution of the model's derivatives influences the epistemic uncertainty OOD (Figure 3). Given that the derivative of a Gaussian process is also a Gaussian process [46], computing the feature Jacobian $\mathbf{J}_{\boldsymbol{\phi}_\mathbf{x}}$ using forward mode AD allows us to reason about the predictive Jacobian in closed-form, which for 1D regression is a vector-valued, probabilistic function, i.e. a Gaussian process, which we denote $\mathbf{z}$,

$$\frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}) := \mathbf{z}(\mathbf{x}) = {\mathbf{J}_{\boldsymbol{\phi}_\mathbf{x}}}^\top \boldsymbol{\beta}, \quad \mathbf{z} \sim p(\cdot \mid \mathbf{x}, \mathcal{D}, \boldsymbol{\theta}), \tag{3}$$

$$p(\mathbf{z} \mid \mathbf{x}, \mathcal{D}, \boldsymbol{\theta}) = \mathcal{GP}(\mathbf{z} | {\mathbf{J}_{\boldsymbol{\phi}_\mathbf{x}}}^\top \boldsymbol{\mu}_n, {\mathbf{J}_{\boldsymbol{\phi}_\mathbf{x}}}^\top \boldsymbol{\Lambda}_n^{-1} \mathbf{J}_{\boldsymbol{\phi}_\mathbf{x}}). \tag{4}$$

In typical regression, $\mathbf{z}$ is unobserved and therefore a quantity we wish to remain uncertain about. Moreover, with expressive function approximators we should be free to shape the uncertainty of $\mathbf{z}$ without interfering with the fit of f. In the Bayesian framework, we can shape this uncertainty by placing a *functional* prior π on $\mathbf{z} \in \mathbb{R}^k$

$$\min_{\boldsymbol{\theta}} D_{\text{KL}}(\pi(\mathbf{z} \mid \mathbf{x}) \,||\, p(\mathbf{z} \mid \mathbf{x}, \mathcal{D}, \boldsymbol{\theta})), \tag{5}$$

enforced through a functional KL divergence (fKL).

Combining the conventional marginal likelihood with this fKL, we propose a novel joint objective

$$\max_{\boldsymbol{\theta}} \ \log p(\mathcal{D} \mid \boldsymbol{\theta}) - D_{\mathrm{KL}}(\pi(\mathbf{z}|\mathbf{x}) \mid\mid p(\mathbf{z} \mid \mathbf{x}, \mathcal{D}, \boldsymbol{\theta})), \quad (6)$$

which can be interpreted from two perspectives.

Maximum Entropy Regularization Since Bayesian linear regression typically considers a fixed feature space or kernel, jointly learning the features can be viewed as an inverse problem [77]. Inverse problems, especially in probabilistic settings, are commonly regularized using the principle of maximum entropy [34, 71]. Choosing the features that are the most unstructured, or the least committed to a specific model, offers not just robustness but ideally translates to calibrated epistemic uncertainty in our setting. One could choose to encourage maximum entropy directly in the predictive distribution. However, optimizing this objective could result in underfitting or increased aleatoric uncertainty, if the data is in conflict with the functional prior. As the derivatives are unobserved, we have more freedom specifying a latent derivative prior. Interestingly, while neural networks are typically overparameterized and benefit from regularization, e.g. weight decay, its role is generally to keep parameters small to avoid overfitting. Due to the Bayesian treatment of the last layer, we are less concerned with overfitting in the features as long as they are sufficiently diverse. The role of the LD prior is to diversify the feature space adequately so that the Bayesian linear model can return a regularized, accurate mean function and expressive predictive variance.

A Latent Variable Model As many regularization schemes can be motivated from a Bayesian reasoning, we can also take a more probabilistic view of the latent derivative term. Given a distribution over latent derivatives, we can construct a latent variable model (LVM) by considering the first-order Taylor expansion (7) over our predictive model f (2). By reparameterizing our regression problem $(y, \mathbf{x})$ into a perturbed form $(y, \bar{\mathbf{x}}, \boldsymbol{\delta})$,

$$y = f(\mathbf{x}) = f(\bar{\mathbf{x}} + \boldsymbol{\delta}) \approx f(\bar{\mathbf{x}}) + \frac{\partial f}{\partial \mathbf{x}}(\bar{\mathbf{x}})^\top \boldsymbol{\delta}, \quad (7)$$

$$= f(\bar{\mathbf{x}}) + \mathbf{z}(\bar{\mathbf{x}})^\top \boldsymbol{\delta}, \quad (8)$$

the above Taylor approximation illustrates how $\mathbf{z}$ influences the predictive uncertainty as the perturbation $\boldsymbol{\delta}$ grows. As typical regression problems only consider directly corresponding pairs $(y, \bar{\mathbf{x}}, \mathbf{0})$, this latent variable perspective is irrelevant for the training data as $\boldsymbol{\delta} = \mathbf{0}$. However, by characterizing prediction between and outside the training data as $\boldsymbol{\delta} \neq \mathbf{0}$, one can appreciate how controlling the distribution of $\mathbf{z}$ influences the epistemic uncertainty in the predictions, as illustrated in Figure 3. This view perhaps helps explain why the the combined objective, Equation (6), strongly resembles the evidence lower bound objective (ELBO) used for inference of LVMs [32]. The key distinction is that $\mathbf{z}$ does not influence the likelihood of the observations, as $\boldsymbol{\delta} = \mathbf{0}$ for the training data. Also, our fKL uses the forward KL (M-projection) instead of the reverse KL (I-projection), which the ELBO uses. The forward KL encourages the distribution to cover as much probability mass as possible which translates to a flat distribution with higher variance, i.e. higher entropy, whereas the reverse KL prefers to seek an individual mode which typically results in lower variance and potential overfitting [47]. We discuss this topic in more detail in Section G of the Appendix.

We now discuss specific aspects of the LDBLL.

The Latent Derivative Objective Although the BLL's derivative distribution can be represented in closed-form, it is a stochastic process rather than a weight distribution. As a result, its KL divergence to a prior process π manifests as a functional KL, which is not a well-defined quantity when the prior does not share the same feature space. In contrast to a regular KL divergence between finite-dimensional probability distributions, a functional KL between stochastic processes requires the evaluation of an infinite-dimensional integral, which is intractable due to the lack of an infinite-dimensional Lebesgue measure [16]. However, it is possible to use a finite index set $\mathcal{T}$ to estimate the otherwise intractable fKL because the fKL between a prior and posterior conditional Gaussian process is equal to the divergence at observations conditioned on $\mathcal{T}$ [76],

$$D_{\mathrm{KL}}(p(\mathbf{f}) \mid\mid p(\mathbf{f} \mid \mathcal{T})) = D_{\mathrm{KL}}(p(\mathbf{f}_{\mathcal{T}}) \mid\mid p(\mathbf{f}_{\mathcal{T}} \mid \mathcal{T})). \quad (9)$$

While we could evaluate the divergence at the training data, i.e. $\mathcal{T}{=}\mathcal{D}$, to account for OOD prediction, we add some noise by defining $\mathcal{T} = \{\mathbf{s}_j \sim \mathcal{N}(\cdot \mid \mathbf{x}_j, \gamma \mathbf{I})\}_{j=1}^n$ and estimate the LD fKL as

$$\frac{1}{|\mathcal{T}|} \sum_{\mathbf{s}_j \in \mathcal{T}} D_{\mathrm{KL}}(\pi(\mathbf{z} \mid \mathbf{s}_j) \mid\mid p(\mathbf{z} \mid \mathbf{s}_j, \mathcal{D}, \boldsymbol{\theta})). \quad (10)$$

The index set $\mathcal{T}$ should ideally represent the true data distribution. Since this data distribution is typically unknown, we create index sets by sampling near the observed training data as a proxy. This is not necessarily the optimal sampling strategy, as this would depend on both the data distribution and task. However, we believe it balances staying within and outside the data distribution, and therefore should be a robust strategy across settings.

Prior Specification We choose the latent derivative prior π as a Gaussian process with a mean function $\boldsymbol{\mu}_\pi$

Table 1: Means and standard errors of test metrics for different BNNs with leaky relu (LR) and tanh (TA) activations for nonlinear regression of gap (Cartpole, CO2, Sarcos, WAM) and standard (UCI) datasets.

(a) Log-Likelihood

		GAP				STANDARD			
MODEL		CARTPOLE	CO2	SARCOS	WAM	BOSTON	CONCRETE	POWER	YACHT
GP	RBF	-4.01 ± 0.00	-4.49 ± 0.00	-5.07 ± 0.03	-2.10 ± 0.01	-2.41 ± 0.06	-3.08 ± 0.02	-2.76 ± 0.01	-0.17 ± 0.03
GBLL	LR	-115.94 ± 50.48	-11.23 ± 1.95	-379.72 ± 53.31	-378.90 ± 41.63	-2.90 ± 0.05	-3.09 ± 0.03	-2.77 ± 0.01	-1.67 ± 0.11
	TA	-27.95 ± 9.96	-8.44 ± 0.92	-403.15 ± 30.66	-173.61 ± 11.61	-3.06 ± 0.03	-3.21 ± 0.03	-2.83 ± 0.01	-0.70 ± 0.10
LDGBLL	LR	-11.68 ± 2.14	-2.04 ± 0.03	-51.98 ± 6.59	-35.36 ± 4.17	-2.60 ± 0.04	-2.97 ± 0.03	-2.77 ± 0.01	-1.13 ± 0.06
	TA	-8.07 ± 1.60	-2.52 ± 0.16	-169.77 ± 5.08	-106.86 ± 8.28	-2.57 ± 0.05	-2.89 ± 0.03	-2.82 ± 0.01	-0.73 ± 0.05
MFVI	LR	-12.19 ± 3.08	-7.23 ± 0.59	-52.23 ± 5.72	-315.55 ± 26.33	-2.39 ± 0.04	-2.97 ± 0.03	-2.77 ± 0.01	-1.43 ± 0.17
	TA	-650.53 ± 358.66	-26.90 ± 1.08	-59.30 ± 4.36	-311.69 ± 19.86	-2.48 ± 0.04	-3.04 ± 0.02	-2.79 ± 0.01	-1.44 ± 0.15
ENSEMBLE	LR	-5.20 ± 0.11	-6.67 ± 0.34	-7.64 ± 0.84	-4.79 ± 0.26	-2.48 ± 0.09	-3.04 ± 0.08	-2.70 ± 0.01	-0.35 ± 0.07
	TA	-3.75 ± 0.28	-9.84 ± 0.41	-13.24 ± 0.83	-17.73 ± 0.88	-2.48 ± 0.08	-3.03 ± 0.07	-2.72 ± 0.01	-0.03 ± 0.05
DROPOUT	LR	-3.73 ± 0.14	-2.42 ± 0.01	-8.58 ± 0.50	-15.46 ± 0.40	-2.36 ± 0.04	-2.90 ± 0.02	-2.80 ± 0.01	-1.82 ± 0.01
	TA	-27.84 ± 1.54	-2.96 ± 0.01	-25.92 ± 0.62	-18.28 ± 0.12	-2.41 ± 0.04	-3.03 ± 0.01	-2.86 ± 0.01	-2.24 ± 0.01
SWAG	LR	-106.72 ± 34.69	-3.56 ± 0.11	-15.34 ± 0.47	-29.49 ± 2.50	-2.64 ± 0.16	-3.19 ± 0.05	-2.77 ± 0.02	-1.11 ± 0.05
MAP	LR	-5800.91 ± 2276.39	-15.73 ± 0.50	-199.49 ± 15.53	-39.54 ± 2.00	-2.60 ± 0.07	-3.04 ± 0.04	-2.77 ± 0.01	-5.14 ± 1.62
	TA	-64.36 ± 21.45	-12.09 ± 0.33	-121.14 ± 10.30	-26.92 ± 0.66	-2.59 ± 0.06	-3.11 ± 0.04	-2.76 ± 0.01	-1.77 ± 0.53

(b) RMSE

		GAP				STANDARD			
MODEL		CARTPOLE	CO2	SARCOS	WAM	BOSTON	CONCRETE	POWER	YACHT
GP	RBF	13.64 ± 0.00	1.70 ± 0.00	2.75 ± 0.00	1.63 ± 0.01	2.83 ± 0.16	5.62 ± 0.13	3.72 ± 0.04	0.40 ± 0.03
GBLL	LR	221.60 ± 55.83	2.53 ± 0.26	3.69 ± 0.15	2.18 ± 0.06	4.19 ± 0.17	5.01 ± 0.18	3.85 ± 0.03	1.09 ± 0.09
	TA	9.47 ± 0.93	2.59 ± 0.17	4.08 ± 0.15	3.26 ± 0.12	4.61 ± 0.23	5.50 ± 0.23	4.09 ± 0.04	0.43 ± 0.03
LDGBLL	LR	179.20 ± 79.00	2.59 ± 0.71	2.80 ± 0.11	1.87 ± 0.06	3.38 ± 0.18	4.80 ± 0.18	3.85 ± 0.04	0.75 ± 0.10
	TA	10.32 ± 1.98	2.38 ± 0.14	2.51 ± 0.03	3.12 ± 0.07	3.12 ± 0.14	4.39 ± 0.14	4.05 ± 0.04	0.52 ± 0.05
MFVI	LR	10.69 ± 2.13	1.82 ± 0.07	2.95 ± 0.19	3.36 ± 0.45	2.74 ± 0.16	4.80 ± 0.13	3.86 ± 0.04	1.10 ± 0.11
	TA	7.72 ± 0.55	3.35 ± 0.11	2.13 ± 0.05	1.46 ± 0.02	2.93 ± 0.13	5.04 ± 0.12	3.91 ± 0.04	1.26 ± 0.14
ENSEMBLE	LR	37.03 ± 4.88	2.10 ± 0.03	3.01 ± 0.05	1.73 ± 0.03	2.79 ± 0.17	4.55 ± 0.12	3.59 ± 0.04	0.83 ± 0.08
	TA	5.50 ± 1.22	2.58 ± 0.03	2.30 ± 0.02	1.36 ± 0.00	2.71 ± 0.13	4.51 ± 0.13	3.66 ± 0.04	0.38 ± 0.03
DROPOUT	LR	4.59 ± 0.22	2.18 ± 0.09	2.67 ± 0.04	1.41 ± 0.02	2.78 ± 0.16	4.45 ± 0.11	3.90 ± 0.04	1.21 ± 0.13
	TA	10.96 ± 0.35	5.19 ± 0.03	2.08 ± 0.02	1.29 ± 0.00	2.77 ± 0.15	4.90 ± 0.10	4.18 ± 0.03	1.20 ± 0.11
SWAG	LR	49.39 ± 8.45	10.73 ± 1.08	3.03 ± 0.07	1.66 ± 0.03	3.08 ± 0.35	5.50 ± 0.16	3.85 ± 0.05	1.13 ± 0.20
MAP	LR	52.50 ± 7.62	1.93 ± 0.03	3.27 ± 0.13	2.04 ± 0.05	3.02 ± 0.17	4.75 ± 0.12	3.81 ± 0.04	0.94 ± 0.09
	TA	6.49 ± 0.62	2.01 ± 0.03	2.67 ± 0.12	1.73 ± 0.02	3.01 ± 0.17	5.15 ± 0.13	3.78 ± 0.04	0.39 ± 0.04

and covariance function $\boldsymbol{\Sigma}_\pi$,

$$\pi(\mathbf{z} \mid \mathbf{x}) = \mathcal{GP}(\mathbf{z} \mid \boldsymbol{\mu}_\pi(\mathbf{x}), \boldsymbol{\Sigma}_\pi(\mathbf{x})). \tag{11}$$

In practice, we set the prior to be constant, with $\boldsymbol{\mu}_\pi(\mathbf{x}) = \mathbf{0}$ and $\boldsymbol{\Sigma}_\pi(\mathbf{x}) = \mathbf{I}$ in whitened data space. The zero mean derivative prior is motivated by the zero mean weight prior. The derivative covariance is harder to specify. While a constant covariance may not be the optimal LD prior for a given task, from a practical perspective it is straightforward to specify, analogous to Gaussian weight priors used for BNNs. Domain knowledge (such as a physics model) could be used to define a more complex derivative prior process, which would combine the benefits of task-specific knowledge and black-box function approximation.

With the Bayesian last layer, it may appear that a LD prior 'overdefines' the BLL and that the two probabilistic treatments conflict. However, as the LD prior seeks to leverage the expressive feature space of the neural network, the universal approximation capability of the neural features should be capable of satisfying both the BLL likelihood and derivative prior. However, due to the construction of the model there are two constraints that can inform our choice of LD prior based on the weight prior. One is that as zero mean weight prior suggests a zero mean derivative prior, due to the linearity of Equation (3). The other is that as $\sigma^2 \to 0$, $\mathbb{V}[\mathbf{z}] \to \mathbf{0}$ due to the weight posterior (defined in Equation (16)) in Equation (4).

In light of this second aspect, we found that scaling the LD prior with the alearotic uncertainty σ^2 improved the prior specification and reduced underfitting in the nonlinear regression tasks. However, the fixed LD prior was beneficial for tasks requiring greater uncertainty quantification, such as active learning. While this scaling can be viewed as a form of empirical Bayes (EB) [48], its limited application suggests better EB approaches may exist. For example, $\boldsymbol{\mu}_\pi$ would benefit from adapting to linear trends in the data, and $\boldsymbol{\Sigma}_\pi$ could be improved by adapting to the relative smoothness w.r.t. each input dimension. We shall investigate alternative approaches in future work. Moreover, this aleatoric scaling arises naturally when considering multivariate output regression and the matrix normal distribution. We discuss the details of this in Appendix A.

4 Experiments

We evaluated the LD prior on several tasks that require predictive uncertainty, namely nonlinear regression, active learning and Bayesian optimization, to verify that our proposed functional latent derivative prior improves the BLL in terms of adequate epistemic uncertainty in the absence of observed data. More detailed discussions and visualizations of all involved datasets can be found in Section I.

4.1 Nonlinear Regression

For the nonlinear regression benchmarks, we compare our LDBLL to the standard BLL and several other baselines: the nonparametric Gaussian process, a regularized network (MAP) and popular BNN approaches. These include mean-field variational inference (MFVI) [7], Monte Carlo dropout [25], ensembles [38] and stochastic weight averaging (SWAG) [44]. All regression problems involve real-world data, however, inspired by previous work based on in-between uncertainty [23], we distinguish between four novel 'gap' tasks, namely Cartpole, CO2, Sarcos and WAM, and 'standard' tasks from the popular UCI benchmark. Our goal is to show that the LDBLL improves the BLL significantly in terms of combating overconfidence during OOD prediction, which shall be demonstrated by the gap tasks, while maintaining competitive performance on the standard benchmarks. Due to the abundance of data, the Gaussian BLL backbone without Bayesian treatment of the observation noise was used for nonlinear regression.
CO2 The Mauna Loa atmospheric carbon dioxide dataset contains CO2 measurements over several decades [63]. To encode the periodicity without using specialized models, we augment the time input with sinusoidal features with an annual frequency. The gap region for testing considers central and edge portions.
Cartpole Here, telemetry is recorded from a Quanser cartpole system performing a swing-up maneuver. We use the dynamic state (position, velocity and acceleration) of the cart and pole for inverse dynamics modeling of the drive torque. The gap region is about the hanging position, where $\theta < 45°$, as depicted by Figure 11 in the Appendix.
Sarcos This dataset [79] contains the telemetry from a 7 DOF manipulator. It is used as a regression benchmark for inverse dynamics modeling, regressing the 21-dimensional state to a drive torque. In the central portion of the data, the robot's pose induces a bias torque (likely due to gravity) in one of the upper motor drives. Therefore, modeling the inverse dynamics on this torque requires OOD prediction. Forecasting unseen aspects of dynamics from limited data represents a key challenge in MBRL for robotics.
WAM This dataset is also derived from a robotic manipulator, the cable-driven 4 DOF Barrett WAM. However, here the distribution shift is generated by demanding the same complex motion at different velocities. By training on a slower motion and evaluating the inverse dynamics model for data collected at a faster speed, the prediction considers the same trajectory but now with higher variance in the values of the state and input due to the larger accelerations at play.
UCI These datasets consists of several disparate regression problems that vary in size and dimension, and are a common benchmark for probabilistic nonlinear regression.
Flight Delay The flight delay dataset is a large-scale regression task of 700k datapoints used to demonstrate scalability [29]. While Bayesian methods are generally less useful for large datasets, as uncertainty should be minimal assuming no distribution shift, models should be able to scale adequately. We detail a batch method for training the BLL using a variational approximation, which aids the model in scaling to large datasets at the cost of non-exact inference during training. We describe this method in Section B and the results in Section I.

More details about the novel gap tasks are discussed in Section I.

Empirical results, displayed in Table 1, show that, in terms of the gap tasks, the LDBLL outperforms the standard BLL significantly in terms of test log-likelihood, which captures the adequacy of the ratio between goodness of fit (RMSE) and predicted uncertainty (entropy). This indicates the LD prior influences a better feature space for predictive uncertainty OOD. For standard regression, results were comparable, which makes sense as OOD uncertainty is not useful in this setting.

With respect to the baselines, the GP, MC dropout and ensembles performed better across gap and standard regression tasks. In fact, the GBLL performance was typically closer to the MAP model than the BNN, and the LD prior did not improve this performance enough to be deemed a competitive alternative. This could be due to capacity (GPs and ensembles have more parameters) and or a superior prior (e.g. the RBF kernel, the Bernoulli weight prior of MC dropout). Superior performance is also characterized by 'underfitting' on the training data (see the tables in Section I.1), sug-

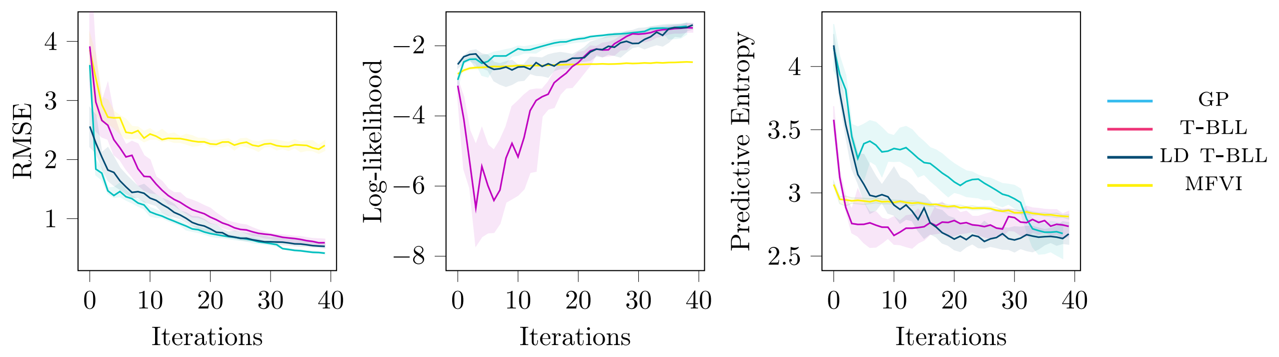

Figure 4: Active learning on the Cartpole dataset, reporting the quartile range over 20 seeds.

gesting the LD prior is not regularizing *enough* during training. While the LD prior is interpretable due to the function space setting, it is not straightforward to assign values to when setting priors for a given task. As empirical Bayes would only tune the prior towards overfitting, it remains an open question how to design the LD prior to provide appropriate regularization for regression tasks.

4.2 Active Learning

Active learning [14] is the setting where a probabilistic model takes an active role in data acquisition, choosing points to optimize learning w.r.t. a utility measure. It is useful in domains such as system identification, where sampling data can be an expensive process. We use the Cartpole dataset introduced in Section 4.1, which contains a dynamical system performing a 'swing-up' control task. The (much smaller) swing-up portion is highly informative while the remaining samples from stabilization are generally redundant. Therefore, information-theoretic data acquisition offers a significant improvement over a random strategy. Section I.2 describes the experiment in detail and Figure 4 shows the results on a held out test set. In this experiment we also compare to a GP, which excels at uncertainty quantification under small datasets. The LDTBLL matches the GP in terms of RMSE and final LLH, however its predictions appear slightly overconfident during learning in comparison. Moreover, the LD prior evidently improves performance significantly on the standard BLL, which is miscalibrated with considerable variance. MFVI struggles to perform the task due to its lackluster uncertainty quantification, which is evident from its relatively small predictive entropy on the test set. As a result, its selected data will likely be collected from uninformative regions and thus essentially random. This would explain its slow progress in both RMSE and LLH improvement.

4.3 Bayesian Optimization

Bayesian optimization (BO) [55, 73] is a sample-efficient black-box global optimization method. By constructing a Bayesian model of the objective, an uncertainty-derived utility function can be used to decide optimal function evaluations. Again, we compare to a GP, which is preferred for BO over BNNs. We performed BO on two tasks (Figure 5), chosen to highlight both the strengths and weaknesses of the LDBLL for BO. They are described in Section I.3

Sinc in a Haystack This toy example is designed to demonstrate the utility of the LDBLL in optimizing high frequency functions, where the optima may be highly local. The function, $f(x) = \mathrm{sinc}(6(x-1))$, is challenging to optimize despite being smooth, as it requires large epistemic uncertainty to avoid suboptimal convergence. While all models demonstrate high variance in performance, the standard TBLL typically fails to achieve any improvement, whereas the LDTBLL is evidently superior. Its 'maximum entropy' nature translates to a powerful exploration strategy.

Hartmann6 This is a standard BO benchmark, with a six-dimensional state and six local minima. Figure 5 shows that the GP is vastly superior at this task, converging rapidly and consistently. This is due in part to the function's smoothness combined with the smoothness assumption of the GP kernel. While the LDBLL converges faster than the BLL, indicating that the LD prior scales to higher dimensions, both converge to a similar suboptimal value compared to the GP. This suggests that either the LDBLL fails to capture the finer grained epistemic uncertainty required to fully converge, or that the specific BO optimizer used here benefits from the GP's smoothness and is less suited to optimizing the BLL due to its increased roughness.

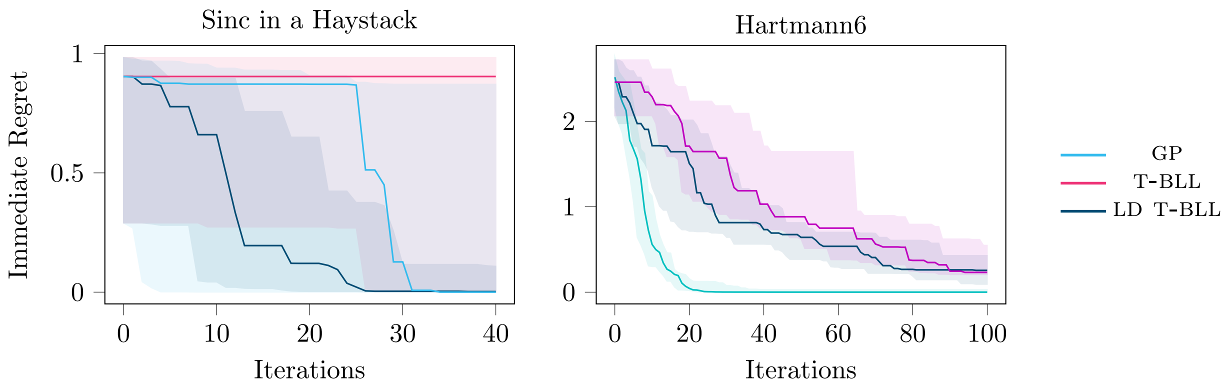

Figure 5: Bayesian optimization on two tasks, displaying the quartiles of regret over 20 seeds.

5 Related Work

Bayesian Neural Networks BNNs have existed since the 1980s as a means of both utilizing neural networks as statistical models [43] and for general regularization [31]. The early work of Neal [50] provided several key contributions, namely the insight that the limit of an infinite hidden layer neural network is a Gaussian process under certain conditions, and the use of Hamiltonian Monte Carlo for approximate inference. Despite the statistical elegance of MCMC training methods [1] and their advancements [33, 12], they are expensive to deploy and scale poorly with larger models. These shortcomings have motivated a focus on variational inference methods [31, 59, 26] for BNNs, including unbiased gradient estimation [7] and other advanced techniques [76, 20, 84, 28]. There is a large family of alternative approximate methods: Including the Laplace approximation [43, 19, 66], ensembles [38, 53, 3, 57], expectation propagation [30], Monte Carlo dropout [25], variational dropout [36], and a range of gradient-based approaches [40, 44]. The Bayesian last layer (also referred to as adaptive basis function and neural linear) model was introduced as a 'marginalized neural network' (MNN) [39] as a neural equivalent to sparse GPs. To mitigate feature overfitting, the MNN uses an ensemble of feature networks. BLLs have previously been applied to Bayesian optimization [74], bandit problems [80, 65], active learning [60], reinforcement learning [52] and regression [51], but there appears a lack of work on improving their general performance. Inference networks [70] are similar to fVI, but take a functional mirror-descent interpretation and incrementally fit the GP prior, enabling minibatch training. Prior Networks [45] use the neural network to parameterize a marginalized distribution, therefore directly predicting Dirichlet distributions for classification.

Gaussian Processes Beyond Neal's infinite limit, there is a rich body of research on the intersection of GPs and NNs. The arcsine (or MLP) [82] and arccosine [13] covariance functions represent the kernel of an infinite single hidden layer network with erf and ReLU activations respectively. The manifold GP [11] uses a neural network to learn an intermediate feature space so that the covariance function performs better on non-smooth functions. Deep kernels [83] define closed-form kernels using neural network components for more expressive covariance functions that are able to incorporate inductive biases such as convolutional operators. Deep Gaussian processes [15, 10, 67] stack GPs to learn a hierarchical representation of intermediate latent variables to build sophisticated statistical models. Moreover, the Student-t process [69] is a Gaussian process with an inverse gamma / Wishart prior over the aleatoric uncertainty.

Functional Priors As Gaussian processes are exact distributions over functions, sparse GPs may be viewed as approximate inference over functions [16], minimizing the fKL from its exact posterior via inducing points. The functional variational BNN (fBNN) [76] uses the fKL to use explicit or implicit stochastic processes as functional priors. They use a GP trained on the data as a prior, which can be viewed as an elaborate form of empirical Bayes. While this prior improves the performance of the variational BNN compared to other methods, it is not evident when and to what extent improvement is made over the GP prior. The noise contrastive prior (NCP) [27] is a similar idea where the training data is perturbed by random noise to serve as a 'data prior' for a BNN, in order to increase uncertainty estimation OOD. While effective empirically, the data prior is again akin to empirical Bayes as the prior is defined by the data. Related work has also considered transforming the BNN weight prior into the prior of a GP [22]. The practice of combining kernels in GPs has been translated to BNN architectures and activation functions, producing periodic and mixing phenomena in the network's feature space for more expressive models [58]. Variational implicit priors [41] use variational inference to worth with functional priors you can only sample from, e.g. simulators, which provides the flexibility of a broad range of complex processes to be adopted as priors.

6 Conclusion

We introduced the latent derivative prior, a novel functional prior for the Bayesian last layer which improves epistemic uncertainty by promoting feature diversity. This model has several attractive properties over weight space BNNs, namely explicit predictive distributions and an intuitive prior that directly enhances functional uncertainty. The LDBLL further demonstrates that, like GPs, *linear* Bayesian models can be sufficient for many problems if the underlying feature space is adequately expressive. We have shown through a suite of tasks that the LD objective significantly improves the uncertainty quantification of the BLL, such that the model is a viable parametric alternative to GPs for downstream tasks like active learning. The LD prior would be further improved by adequate specification for a given task or dataset. Using the notion of derivatives, this prior could provide a way of incorporating domain knowledge (i.e. from physics) into the model to improve performance over pure black box models. Moreover, the application of the BLL and LDBLL to multivariate prediction tasks such as model-based control and classification is an open avenue, in particular how the notion of derivative uncertainty applies to classification.

Acknowledgements

We wish to thank Svenja Stark and Michael Lutter for proofreading and feedback. Pascal Klink and Joni Pajarinen are funded by the DFG project PA3179/1-1 (ROBOLEAP). Furthermore, this research was supported by grants from NVIDIA and the NVIDIA DGX Station. Thanks also to Danijar Hanfner for providing access to the flight delay dataset and the anonymous reviewers for helpful comments during the review process.

References

[1] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*, 2003.

[2] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020.

[3] D. Barber and Christopher Bishop. Ensemble learning in bayesian neural networks. In *Generalization in Neural Networks and Machine Learning*, 1998.

[4] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. The curse of dimensionality for local kernel machines. Technical Report TR-1258, Université de Montréal, 2005.

[5] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*, 2018.

[6] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer-Verlag, Berlin, Heidelberg, 2006.

[7] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, 2015.

[8] Alexandr A. Borovkov. *Probability Theory.* Springer London, 2013.

[9] G. E. P. Box and G. C. Tiao. *Bayesian Inference in Statistical Analysis.* John Wiley & Sons, New York, 1973.

[10] Thang Bui, Daniel Hernández-Lobato, Jose Hernandez-Lobato, Yingzhen Li, and Richard Turner. Deep gaussian processes for regression using approximate expectation propagation. In *International Conference on Machine Learning*, 2016.

[11] R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth. Manifold gaussian processes for regression. In *International Joint Conference on Neural Networks*, 2016.

[12] Tianqi Chen, Emily B. Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International Conference on Machine Learning*, 2014.

[13] Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems*, 2009.

[14] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. In *Advances in Neural Information Processing Systems*, 1995.

[15] Andreas Damianou and Neil Lawrence. Deep gaussian processes. In *Artificial Intelligence and Statistics*, 2013.

[16] Alexander G. de G. Matthews, James Hensman, Richard Turner, and Zoubin Ghahramani. On sparse variational methods and the kullback-leibler divergence between stochastic processes. In *International Conference on Artificial Intelligence and Statistics*, 2016.

[17] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning*, 2011.

[18] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2013.

[19] John S. Denker and Yann LeCun. Transforming neural-net output levels to probability distributions. In *Advances in Neural Information Processing Systems*, 1991.

[20] M. Emtiyaz Khan, D. Nielsen, V. Tangkaratt, W. Lin, Y. Gal, and A. Srivastava. Fast and Scalable Bayesian Deep Learning by Weight-Perturbation in Adam. In *International Conference on Machine Learning*, 2018.

[21] Angelos Filos, Sebastian Farquhar, Aidan N. Gomez, Tim G. J. Rudner, Zachary Kenton, Lewis Smith, Milad Alizadeh, Arnoud de Kroon, and Yarin Gal. Benchmarking bayesian deep learning with diabetic retinopathy diagnosis. `https://github.com/OATML/bdl-benchmarks`, 2019.

[22] Daniel Flam-Shepherd, James Requeima, and David Duvenaud. Mapping gaussian process priors to bayesian neural networks. In *NIPS Bayesian deep learning workshop*, 2017.

[23] Andrew Foong, Yingzhen Li, José Hernández-Lobato, and Richard Turner. 'in-between' uncertainty in bayesian neural networks. In *ICML Workshop on Uncertainty and Robustness in Deep Learning*, 2019.

[24] Andrew Y. K. Foong, David R. Burt, Yingzhen Li, and Richard E. Turner. On the expressiveness of approximate inference in bayesian neural networks. *arxiv e-prints*, 2019.

[25] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, 2016.

[26] Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, 2011.

[27] Danijar Hafner, Dustin Tran, Alex Irpan, Timothy Lillicrap, and James Davidson. Noise contrastive priors for functional uncertainty. In *Uncertainty in Artificial Intelligence*, 2019.

[28] Manuel Haußmann, Fred A. Hamprecht, and M. Kandemir. Sampling-free variational inference of bayesian neural networks by variance backpropagation. In *Uncertainty in Artificial Intelligence*, 2019.

[29] James Hensman, Nicolò Fusi, and Neil D. Lawrence. Gaussian processes for big data. In *Uncertainty in Artificial Intelligence*, 2013.

[30] José Miguel Hernández-Lobato and Ryan P. Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, 2015.

[31] Geoffrey E. Hinton and Drew van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Conference on Computational Learning Theory*, 1993.

[32] Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(4):1303–1347, 2013.

[33] Matthew D. Homan and Andrew Gelman. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15, 2014.

[34] E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106, 1957.

[35] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[36] Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, 2015.

[37] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M Blei. Automatic differentiation variational inference. *The Journal of Machine Learning Research*, 2017.

[38] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, 2017.

[39] Miguel Lázaro-Gredilla and Aníbal R. Figueiras-Vidal. Marginalized neural network mixtures for large-scale regression. *Transactions on Neural Networks*, 21(8), 2010.

[40] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances in Neural Information Processing Systems*, 2016.

[41] Chao Ma, Yingzhen Li, and Jose Miguel Hernandez-Lobato. Variational implicit processes. In *International Conference on Machine Learning*, 2019.

[42] David J. C. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4), 1992.

[43] David J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3), 1992.

[44] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, 2019.

[45] Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems*, 2018.

[46] Andrew McHutchon. *Nonlinear Modelling and Control using Gaussian Processes.* PhD thesis, University of Cambridge, 2014.

[47] Tom Minka et al. Divergence measures and message passing. Technical report, Technical report, Microsoft Research, 2005.

[48] Carl N Morris. Parametric empirical bayes inference: theory and applications. *Journal of the American statistical Association*, 78(381), 1983.

[49] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective.* The MIT Press, 2012.

[50] Radford M. Neal. *Bayesian Learning for Neural Networks.* PhD thesis, University of Toronto, CAN, 1995.

[51] Sebastian W. Ober and Carl Edward Rasmussen. Benchmarking the neural linear model for regression. In *Symposium on Advances in Approximate Bayesian Inference*, 2019.

[52] Brendan O'Donoghue, Ian Osband, Rémi Munos, and Volodymyr Mnih. The uncertainty bellman equation and exploration. In Jennifer G. Dy and Andreas Krause, editors, *International Conference on Machine Learning*, 2018.

[53] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2018.

[54] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, 2019.

[55] Anthony O'Hagan. Some bayesian numerical analysis. *Bayesian Statistics*, 1992.

[56] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019.

[57] Tim Pearce, Felix Leibfried, Alexandra Brintrup, Mohamed Zaki, and Andy Neely. Uncertainty in neural networks: Approximately bayesian ensembling. In *International Conference on Artificial Intelligence and Statistics*, 2020.

[58] Tim Pearce, Russell Tsuchida, Mohamed Zaki, Alexandra Brintrup, and Andy Neely. Expressive priors in bayesian neural networks: Kernel combinations and periodic functions. In *Uncertainty in Artificial Intelligence*, 2019.

[59] Carsten Peterson and Eric Hartman. Explorations of the mean field theory learning algorithm. *Neural Networks*, 1989.

[60] Robert Pinsler, Jonathan Gordon, Eric Nalisnick, and José Miguel Hernández-Lobato. Bayesian batch active learning as sparse subset approximation. In *Advances in Neural Information Processing Systems*, 2019.

[61] Louis B. Rall. *Automatic Differentiation: Techniques and Applications*, volume 120 of *Lecture Notes in Computer Science*. Springer, 1981.

[62] Carl Edward Rasmussen and Zoubin Ghahramani. Occam's razor. In *Advances in Neural Information Processing Systems*, 2001.

[63] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

[64] CE. Rasmussen. Gaussian processes to speed up hybrid monte carlo for expensive bayesian integrals. *Bayesian Statistics*, 2003.

[65] Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. In *International Conference on Learning Representations*, 2018.

[66] Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *International Conference on Learning Representations*, 2018.

[67] Hugh Salimbeni and Marc Deisenroth. Doubly stochastic variational inference for deep gaussian processes. In *Advances in Neural Information Processing Systems*, 2017.

[68] Matthias Seeger. *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximation*. PhD thesis, University of Edinburgh, 2003.

[69] Amar Shah, Andrew Wilson, and Zoubin Ghahramani. Student-t Processes as Alternatives to Gaussian Processes. In *International Conference on Artificial Intelligence and Statistics*, 2014.

[70] Jiaxin Shi, Mohammad Emtiyaz Khan, and Jun Zhu. Scalable training of inference networks for Gaussian-process models. In *International Conference on Machine Learning*, 2019.

[71] C Ray Smith and Walter T Grandy Jr. *Maximum-Entropy and bayesian methods in inverse problems*, volume 14. Springer Science & Business Media, 2013.

[72] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, 2005.

[73] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, 2012.

[74] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Md. Mostofa Ali Patwary, Prabhat Prabhat, and Ryan P. Adams. Scalable bayesian optimization using deep neural networks. In *International Conference on Machine Learning*, 2015.

[75] E. Solak, R. Murray-Smith, W.E. Leithead, D.J. Leith, and C.E. Rasmussen. Derivative observations in gaussian process models of dynamic systems. In *Advances in Neural Information Processing Systems*, 2003.

[76] Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional variational bayesian neural networks. In *International Conference on Learning Representations*, 2019.

[77] Albert Tarantola. *Inverse problem theory and methods for model parameter estimation*, volume 89. siam, 2005.

[78] Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, 2009.

[79] S. Vijayakumar and S. Schaal. Locally weighted projection regression: An o(n) algorithm for incremental real time learning in high dimensional spaces. In *International Conference on Machine Learning*, 2000.

[80] Noah Weber, Janez Starc, Arpit Mittal, Roi Blanco, and Lluís Màrquez. Optimizing over a bayesian last layer. In *NeurIPS Bayesian Deep Learning Workshop*, 2018.

[81] Florian Wenzel, Kevin Roth, Bastiaan S Veeling, Jakub Świątkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the bayes posterior in deep neural networks really? In *International Conference on Machine Learning*, 2020.

[82] Christopher K. I. Williams. Computing with infinite networks. In *Advances in Neural Information Processing Systems*, 1996.

[83] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. In *International Conference on Artificial Intelligence and Statistics*, 2016.

[84] Anqi Wu, Sebastian Nowozin, Ted Meeds, Richard E. Turner, Jose Miguel Hernadez-Lobato, and Alexander L. Gaunt. Deterministic variational inference for robust bayesian neural networks. In *International Conference on Learning Representations*, 2019.

[85] J. Yao, W. Pan, S. Ghosh, and F. Doshi-Velez. Quality of uncertainty quantification for bayesian neural network inference. In *ICML Workshop on Uncertainty & Robustness in Deep Learning*, 2019.

A Bayesian Last Layer Equations

In this section, we collectively state the equations for Bayesian linear regression [9, 6, 49] with weights $\boldsymbol{\beta}$ and additive noise ϵ, where

$$y_i = f(\mathbf{x}_i; \boldsymbol{\theta}) = \boldsymbol{\phi}_i^\top \boldsymbol{\beta} + \epsilon_i, \tag{12}$$

$$\epsilon_i \sim \mathcal{N}(\cdot \mid 0, \sigma^2), \tag{13}$$

$$\mathbf{y} \sim \mathcal{N}(\cdot \mid \boldsymbol{\Phi\beta}, \sigma^2 \mathbf{I}). \tag{14}$$

Assuming known aleatoric uncertainty σ^2 [6], a conjugate Gaussian prior over $\boldsymbol{\beta}$ results in a Gaussian posterior,

$$\boldsymbol{\beta} \sim \mathcal{N}(\cdot \mid \boldsymbol{\mu}_0, \boldsymbol{\Lambda}_0^{-1}), \qquad \boldsymbol{\mu}_n = \boldsymbol{\Lambda}_n^{-1}(\boldsymbol{\Lambda}_0 \boldsymbol{\mu}_0 + \sigma^{-2} \boldsymbol{\Phi}^\top \mathbf{y}), \tag{15}$$

$$\boldsymbol{\beta} \mid \mathcal{D}, \boldsymbol{\theta} \sim \mathcal{N}(\cdot \mid \boldsymbol{\mu}_n, \boldsymbol{\Lambda}_n^{-1}), \qquad \boldsymbol{\Lambda}_n = \sigma^{-2} \boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \boldsymbol{\Lambda}_0, \tag{16}$$

with an explicit Gaussian predictive distribution for output y and query $\mathbf{x}$,

$$y \mid \mathbf{x}, \mathcal{D}, \boldsymbol{\theta} \sim \mathcal{N}(\cdot \mid \boldsymbol{\phi}_\mathbf{x}^\top \boldsymbol{\mu}_n, \sigma^2 + \boldsymbol{\phi}_\mathbf{x}^\top \boldsymbol{\Lambda}_n^{-1} \boldsymbol{\phi}_\mathbf{x}). \tag{17}$$

The log-marginal likelihood can be written as

$$\log p(\mathcal{D} \mid \boldsymbol{\theta}) = \frac{1}{2}(\boldsymbol{\mu}_n^\top \boldsymbol{\Lambda}_n \boldsymbol{\mu}_n - \boldsymbol{\mu}_0^\top \boldsymbol{\Lambda}_0 \boldsymbol{\mu}_0) - \frac{1}{2\sigma^2}\mathbf{y}^\top \mathbf{y} - \frac{n}{2}\log 2\pi\sigma^2 + \frac{1}{2}\log|\boldsymbol{\Lambda}_0| - \frac{1}{2}\log|\boldsymbol{\Lambda}_n|. \tag{18}$$

Assuming unknown aleatoric uncertainty σ^2 [49], a conjugate normal-inverse-gamma prior over $\boldsymbol{\beta}$ and σ^2, such that the joint prior distribution of $\boldsymbol{\beta}$ and σ^2 factorizes as $p(\boldsymbol{\beta}, \sigma^2) = p(\boldsymbol{\beta} \mid \sigma^2) p(\sigma^2)$, where

$$\boldsymbol{\beta} \mid \sigma^2 \sim \mathcal{N}(\cdot \mid \boldsymbol{\mu}_0, \sigma^2 \boldsymbol{\Lambda}_0^{-1}), \tag{19}$$

$$\sigma^2 \sim \text{Inv-Gamma}(\cdot \mid a_0, b_0), \tag{20}$$

results in a joint posterior distribution which factorizes as $p(\boldsymbol{\beta}, \sigma^2 \mid \mathcal{D}, \boldsymbol{\theta}) = p(\boldsymbol{\beta} \mid \sigma^2, \mathcal{D}, \boldsymbol{\theta}) p(\sigma^2 \mid \mathcal{D}, \boldsymbol{\theta})$, where

$$\boldsymbol{\beta} \mid \sigma^2, \mathcal{D}, \boldsymbol{\theta} \sim \mathcal{N}(\cdot \mid \boldsymbol{\mu}_n, \sigma^2 \boldsymbol{\Lambda}_n^{-1}), \qquad \sigma^2 \mid \mathcal{D}, \boldsymbol{\theta} \sim \text{Inv-Gamma}(\cdot \mid a_n, b_n), \tag{21}$$

$$\boldsymbol{\mu}_n = \boldsymbol{\Lambda}_n^{-1}(\boldsymbol{\Lambda}_0 \boldsymbol{\mu}_0 + \boldsymbol{\Phi}^\top \mathbf{y}), \qquad a_n = a_0 + \tfrac{n}{2}, \tag{22}$$

$$\boldsymbol{\Lambda}_n = \boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \boldsymbol{\Lambda}_0, \qquad b_n = b_0 + \tfrac{1}{2}(\mathbf{y}^\top \mathbf{y} + \boldsymbol{\mu}_0^\top \boldsymbol{\Lambda}_0 \boldsymbol{\mu}_0 - \boldsymbol{\mu}_n^\top \boldsymbol{\Lambda}_n \boldsymbol{\mu}_n). \tag{23}$$

The marginal posterior for $\boldsymbol{\beta}$ is obtained by integrating $p(\boldsymbol{\beta}, \sigma^2 \mid \mathcal{D}, \boldsymbol{\theta})$ over σ^2, resulting in a Student's t-distribution with $2a_n$ degrees of freedom,

$$\boldsymbol{\beta} \mid \mathcal{D}, \boldsymbol{\theta} \sim \text{St}(\cdot \mid \boldsymbol{\mu}_n, \tfrac{b_n}{a_n} \boldsymbol{\Lambda}_n^{-1}, 2a_n). \tag{24}$$

The predictive distribution is also a Student's t-distribution with $2a_n$ degrees of freedom,

$$y \mid \mathbf{x}, \mathcal{D}, \boldsymbol{\theta} \sim \text{St}(\cdot \mid \boldsymbol{\phi}_\mathbf{x}^\top \boldsymbol{\mu}_n, \tfrac{b_n}{a_n}(1 + \boldsymbol{\phi}_\mathbf{x}^\top \boldsymbol{\Lambda}_n^{-1} \boldsymbol{\phi}_\mathbf{x}), 2a_n). \tag{25}$$

The log-marginal likelihood can be written as

$$\log p(\mathcal{D} \mid \boldsymbol{\theta}) = \log \Gamma(a_n) - \log \Gamma(a_0) + a_0 \log b_0 - a_n \log b_n + \frac{1}{2}\log|\boldsymbol{\Lambda}_0| - \frac{1}{2}\log|\boldsymbol{\Lambda}_n| - \frac{n}{2}\log 2\pi, \tag{26}$$

where Γ denotes the gamma function.

B Scalable Batch Training using Variational Inference

While the BLL can be trained using exact inference, this results in gradient updates that leverage the entire training dataset each iteration. This poses a scaling issue for large models and datasets due to the memory requirements during backpropagation, especially in the case of a LD prior as the Jacobians are also computed and stored. Previous work have avoided this complexity by using the MAP approximation while training the features [74], however this discards the Bayesian component, which we require for fVI. The closed-form posteriors can also be computed using sequential Bayesian updates on batches of data [6], however we found that this dramatically increases the complexity of the computation graph and rendered backpropagation for the features unfeasibly slow for large models.

To offer a compromise between inference accuracy and scalable learning, we detail a training scheme based on automatic differentiation variational inference (adVI) [37], where the posterior updates and log-marginal likelihood objective is replaced by the ELBO. Optimizing the (exact) variational posterior, inference and feature learning can be performed using backpropagation on batches of data. Since the expected likelihood term of the ELBO can be computed exactly, we still retain some of the benefits of the BLL approach. Defining a variational posterior $q_\phi(\boldsymbol{\beta} \mid \mathcal{D}) = \mathcal{N}(\boldsymbol{\mu}_q, \boldsymbol{\Lambda}_q^{-1})$ with parameters ϕ, the ELBO objective (that replaces the log marginal) is

$$\mathcal{L}_{\text{ELBO}}(\mathcal{D}_{\text{batch}}, \phi) = \mathbb{E}_{q_\phi(\boldsymbol{\beta}|\mathcal{D})}[\log p(\mathcal{D}_{\text{batch}} \mid \boldsymbol{\beta})] - \frac{n_{\text{batch}}}{n} D_{\text{KL}}(q_\phi(\boldsymbol{\beta} \mid \mathcal{D}) \,||\, p(\boldsymbol{\beta})). \tag{27}$$

For univariate prediction where $\boldsymbol{\beta}$ is the multivariate normal, the expected loglikelihood is

$$\mathbb{E}_{q_\phi(\boldsymbol{\beta}|\mathcal{D})}[\log p(\mathcal{D} \mid \boldsymbol{\beta})] = -\frac{n}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}(\mathbf{y}^\top\mathbf{y} - \mathbf{y}^\top\Phi\boldsymbol{\mu}_q - \boldsymbol{\mu}_q^\top\Phi^\top\mathbf{y} + \boldsymbol{\mu}_q^\top\Phi^\top\Phi\boldsymbol{\mu}_q + \text{tr}\{\Phi^\top\Phi\boldsymbol{\Lambda}_q^{-1}\}). \tag{28}$$

After training, we compute the exact posterior for the learned features using sequential Bayes on minibatches of data. Using this variation, we observe that on real data there is typically a reduction in performance, indicating features learned using exact inference are superior to those from this variational approximation. The main weakness is that empirical Bayes is no longer easy to perform, as it would now require expensive bilevel optimization of the prior and ELBO. Not optimizing the weight prior may result in underfitting for certain tasks. For the implementation, $\boldsymbol{\Lambda}_q$ was parameterized through a diagonal and lower triangular matrix to ensure positive definiteness. Note that when the variational posterior is the true posterior (Equations 15-16), the ELBO objective is equal to the log-marginal likelihood (Equation (18)) by definition.

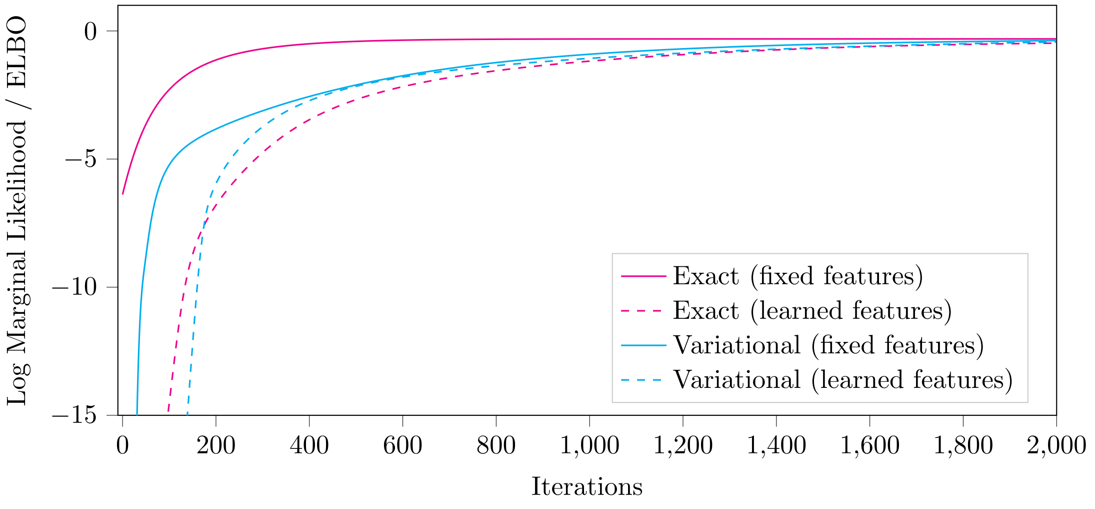

Figure 6: Illustration of the performance gap between the exact and variational BLL on a toy 1D regression problem. While the weight prior was fixed, the aleatoric variance was optimized for all cases. As expected, the variational approximation lags behind exact inference, however when combined with feature learning the variational objective aids convergence, presumably because it is a numerically beneficial objective. Reassuringly, all models converge to similar performance. For the fixed feature space, fourier features of increasing frequency were used. For the learned features, two layers of tanh activations were used. The width of both feature spaces were equal.

C Bayesian Last Layer Derivative Distribution

To compute the derivative of the predictive random variable y, we apply the limit definition of the derivative to two distinct predictions without aleatoric noise, namely $f(\boldsymbol{\xi}) = \bar{f}_{\boldsymbol{\xi}} + \epsilon_{\boldsymbol{\xi}}$ and $f(\boldsymbol{\xi}+\boldsymbol{\delta}) = \bar{f}_{\boldsymbol{\delta}} + \epsilon_{\boldsymbol{\delta}}$, where $\bar{f}$ is the predictive mean function and ϵ is the zero-centered epistemic uncertainty (17, 25)

$$\frac{\partial f}{\partial \mathbf{x}}(\boldsymbol{\xi}) = \lim_{\boldsymbol{\delta}\to\mathbf{0}} \frac{f(\boldsymbol{\xi}+\boldsymbol{\delta}) - f(\boldsymbol{\xi})}{\boldsymbol{\xi}+\boldsymbol{\delta}-\boldsymbol{\xi}}, \tag{29}$$

$$= \lim_{\boldsymbol{\delta}\to\mathbf{0}} \frac{\bar{f}_{\boldsymbol{\delta}} + \epsilon_{\boldsymbol{\delta}} - \bar{f}_{\boldsymbol{\xi}} - \epsilon_{\boldsymbol{\xi}}}{\boldsymbol{\delta}}, \tag{30}$$

$$= \lim_{\boldsymbol{\delta}\to\mathbf{0}} \frac{\bar{f}_{\boldsymbol{\delta}} - \bar{f}_{\boldsymbol{\xi}}}{\boldsymbol{\delta}} + \lim_{\boldsymbol{\delta}\to\mathbf{0}} \frac{\epsilon_{\boldsymbol{\delta}} - \epsilon_{\boldsymbol{\xi}}}{\boldsymbol{\delta}}. \tag{31}$$

Here, the first term corresponds to its expected value and the second term represents its variance. The following derivations make use of the Uniform Convergence Theorem (UCT) to change the order of expectations, variances and limits [8].

To compute the expected value, we rearrange until we can apply the expectation operator to the individual terms. Afterwards, we separate the terms which are relevant for the limit from the terms which are constant w.r.t. the limit. Finally, we evaluate the limit

$$\mathbb{E}\left[\frac{\partial f}{\partial \mathbf{x}}(\boldsymbol{\xi})\right], = \mathbb{E}\left[\lim_{\boldsymbol{\delta}\to\mathbf{0}} \frac{f(\boldsymbol{\xi}+\boldsymbol{\delta}) - f(\boldsymbol{\xi})}{\boldsymbol{\xi}+\boldsymbol{\delta}-\boldsymbol{\xi}}\right], \tag{32}$$

$$= \mathbb{E}\left[\lim_{\boldsymbol{\delta}\to\mathbf{0}} \frac{\bar{f}_{\boldsymbol{\delta}} + \epsilon_{\boldsymbol{\delta}} - \bar{f}_{\boldsymbol{\xi}} - \epsilon_{\boldsymbol{\xi}}}{\boldsymbol{\delta}}\right], \tag{33}$$

$$= \lim_{\boldsymbol{\delta}\to\mathbf{0}} \frac{\mathbb{E}\left[\bar{f}_{\boldsymbol{\delta}}\right] + \mathbb{E}\left[\epsilon_{\boldsymbol{\delta}}\right] - \mathbb{E}\left[\bar{f}_{\boldsymbol{\xi}}\right] - \mathbb{E}\left[\epsilon_{\boldsymbol{\xi}}\right]}{\boldsymbol{\delta}}, \tag{34}$$

$$= \lim_{\boldsymbol{\delta}\to\mathbf{0}} \frac{\boldsymbol{\phi}_{\boldsymbol{\delta}}^\top \boldsymbol{\mu}_n + \mathbf{0} - \boldsymbol{\phi}_{\boldsymbol{\xi}}^\top \boldsymbol{\mu}_n - \mathbf{0}}{\boldsymbol{\delta}}, \tag{35}$$

$$= \left[\lim_{\boldsymbol{\delta}\to\mathbf{0}} \frac{\boldsymbol{\phi}_{\boldsymbol{\delta}} - \boldsymbol{\phi}_{\boldsymbol{\xi}}}{\boldsymbol{\delta}}\right]^\top \boldsymbol{\mu}_n, \tag{36}$$

$$= \mathbf{J}_{\boldsymbol{\phi}_{\boldsymbol{\xi}}}^\top \boldsymbol{\mu}_n. \tag{37}$$

Here, $\mathbf{J}_{\boldsymbol{\phi}_{\boldsymbol{\xi}}}$ represents the Jacobian of $\boldsymbol{\phi}(\cdot;\boldsymbol{\theta})$ evaluated at $\boldsymbol{\xi}$. Since the predictive mean functions for both the Gaussian and the Student-t models are the same, the expected value of their derivative distributions are also the same.

Before we compute the variance, we first derive a closed-form expression for the joint zero-centered epistemic uncertainty $p(\epsilon_{\boldsymbol{\xi}}, \epsilon_{\boldsymbol{\delta}})$, which we will need later. Subtracting the mean and discarding the aleatoric noise component from the predictive distributions (17, 25) yield

$$p(\epsilon_{\boldsymbol{\xi}}, \epsilon_{\boldsymbol{\delta}}) = \mathcal{N}\left(\begin{bmatrix}\epsilon_{\boldsymbol{\xi}} \\ \epsilon_{\boldsymbol{\delta}}\end{bmatrix} \middle| \mathbf{0}, \begin{bmatrix}\boldsymbol{\phi}_{\boldsymbol{\xi}}^\top \boldsymbol{\Lambda}_n^{-1} \boldsymbol{\phi}_{\boldsymbol{\xi}} & \boldsymbol{\phi}_{\boldsymbol{\xi}}^\top \boldsymbol{\Lambda}_n^{-1} \boldsymbol{\phi}_{\boldsymbol{\delta}} \\ \boldsymbol{\phi}_{\boldsymbol{\delta}}^\top \boldsymbol{\Lambda}_n^{-1} \boldsymbol{\phi}_{\boldsymbol{\xi}} & \boldsymbol{\phi}_{\boldsymbol{\delta}}^\top \boldsymbol{\Lambda}_n^{-1} \boldsymbol{\phi}_{\boldsymbol{\delta}}\end{bmatrix}\right), \tag{38}$$

for the Gaussian and

$$p(\epsilon_{\boldsymbol{\xi}}, \epsilon_{\boldsymbol{\delta}}) = \mathrm{St}\left(\begin{bmatrix}\epsilon_{\boldsymbol{\xi}} \\ \epsilon_{\boldsymbol{\delta}}\end{bmatrix} \middle| \mathbf{0}, \frac{b_n}{a_n}\begin{bmatrix}\boldsymbol{\phi}_{\boldsymbol{\xi}}^\top \boldsymbol{\Lambda}_n^{-1} \boldsymbol{\phi}_{\boldsymbol{\xi}} & \boldsymbol{\phi}_{\boldsymbol{\xi}}^\top \boldsymbol{\Lambda}_n^{-1} \boldsymbol{\phi}_{\boldsymbol{\delta}} \\ \boldsymbol{\phi}_{\boldsymbol{\delta}}^\top \boldsymbol{\Lambda}_n^{-1} \boldsymbol{\phi}_{\boldsymbol{\xi}} & \boldsymbol{\phi}_{\boldsymbol{\delta}}^\top \boldsymbol{\Lambda}_n^{-1} \boldsymbol{\phi}_{\boldsymbol{\delta}}\end{bmatrix}, 2a_n\right), \tag{39}$$

for the Student-t model, respectively.

To compute the variance, we follow a similar procedure as for the expected value, first rearranging and then applying the variance operator to the individual terms using the variance rule for the sum of two random variables. Since the predictive mean is constant w.r.t. the variance operator, the corresponding terms evaluate to zero.

Thus, the variance operator is only applied to ϵ

$$\begin{aligned}
\mathbb{V}\left[\frac{\partial f}{\partial \mathbf{x}}(\boldsymbol{\xi})\right] &= \mathbb{V}\left[\lim_{\boldsymbol{\delta}\to\mathbf{0}} \frac{f(\boldsymbol{\xi}+\boldsymbol{\delta}) - f(\boldsymbol{\xi})}{\boldsymbol{\xi}+\boldsymbol{\delta}-\boldsymbol{\xi}}\right], && (40)\\
&= \mathbb{V}\left[\lim_{\boldsymbol{\delta}\to\mathbf{0}} \frac{\bar{f}_{\boldsymbol{\delta}} + \epsilon_{\boldsymbol{\delta}} - \bar{f}_{\boldsymbol{\xi}} - \epsilon_{\boldsymbol{\xi}}}{\boldsymbol{\delta}}\right], && (41)\\
&= \lim_{\boldsymbol{\delta}\to\mathbf{0}} \frac{1}{\boldsymbol{\delta}^2}\left(\mathbb{V}\left[\bar{f}_{\boldsymbol{\delta}}+\epsilon_{\boldsymbol{\delta}}\right] + \mathbb{V}\left[\bar{f}_{\boldsymbol{\xi}}+\epsilon_{\boldsymbol{\xi}}\right] - \mathbb{C}[\bar{f}_{\boldsymbol{\delta}}+\epsilon_{\boldsymbol{\delta}}, \bar{f}_{\boldsymbol{\xi}}+\epsilon_{\boldsymbol{\xi}}] - \mathbb{C}[\bar{f}_{\boldsymbol{\xi}}+\epsilon_{\boldsymbol{\xi}}, \bar{f}_{\boldsymbol{\delta}}+\epsilon_{\boldsymbol{\delta}}]\right), && (42)\\
&= \lim_{\boldsymbol{\delta}\to\mathbf{0}} \frac{1}{\boldsymbol{\delta}^2}\left(\mathbb{V}[\epsilon_{\boldsymbol{\delta}}]+\mathbb{V}[\epsilon_{\boldsymbol{\xi}}] - \mathbb{C}[\epsilon_{\boldsymbol{\delta}}, \epsilon_{\boldsymbol{\xi}}] - \mathbb{C}[\epsilon_{\boldsymbol{\xi}}, \epsilon_{\boldsymbol{\delta}}]\right). && (43)
\end{aligned}$$

Replacing the variance terms with the Gaussian model yields

$$\begin{aligned}
\mathbb{V}\left[\frac{\partial f}{\partial \mathbf{x}}(\boldsymbol{\xi})\right] &= \lim_{\boldsymbol{\delta}\to\mathbf{0}} \frac{1}{\boldsymbol{\delta}^2}\left(\boldsymbol{\phi}_{\boldsymbol{\delta}}^\top \boldsymbol{\Lambda}_n^{-1}\boldsymbol{\phi}_{\boldsymbol{\delta}} + \boldsymbol{\phi}_{\boldsymbol{\xi}}^\top \boldsymbol{\Lambda}_n^{-1}\boldsymbol{\phi}_{\boldsymbol{\xi}} - \boldsymbol{\phi}_{\boldsymbol{\xi}}^\top \boldsymbol{\Lambda}_n^{-1}\boldsymbol{\phi}_{\boldsymbol{\delta}} - \boldsymbol{\phi}_{\boldsymbol{\delta}}^\top \boldsymbol{\Lambda}_n^{-1}\boldsymbol{\phi}_{\boldsymbol{\xi}}\right), && (44)\\
&= \lim_{\boldsymbol{\delta}\to\mathbf{0}} \frac{1}{\boldsymbol{\delta}^2}\left((\boldsymbol{\phi}_{\boldsymbol{\delta}} - \boldsymbol{\phi}_{\boldsymbol{\xi}})^\top \boldsymbol{\Lambda}_n^{-1}(\boldsymbol{\phi}_{\boldsymbol{\delta}} - \boldsymbol{\phi}_{\boldsymbol{\xi}})\right), && (45)\\
&= \mathbf{J}_{\boldsymbol{\phi}_{\boldsymbol{\xi}}}^\top \boldsymbol{\Lambda}_n^{-1}\mathbf{J}_{\boldsymbol{\phi}_{\boldsymbol{\xi}}}. && (46)
\end{aligned}$$

The corresponding derivation for the Student-t model is almost identical with the only difference being a scalar factor. Note that the Student-t distribution requires an additional factor to convert from scale into variance, which we will drop again later to convert back into scale

$$\begin{aligned}
\mathbb{V}\left[\frac{\partial f}{\partial \mathbf{x}}(\boldsymbol{\xi})\right] &= \frac{2a_n}{2a_n-2}\frac{b_n}{a_n} \lim_{\boldsymbol{\delta}\to\mathbf{0}} \frac{1}{\boldsymbol{\delta}^2}\left(\boldsymbol{\phi}_{\boldsymbol{\delta}}^\top \boldsymbol{\Lambda}_n^{-1}\boldsymbol{\phi}_{\boldsymbol{\delta}} + \boldsymbol{\phi}_{\boldsymbol{\xi}}^\top \boldsymbol{\Lambda}_n^{-1}\boldsymbol{\phi}_{\boldsymbol{\xi}} - \boldsymbol{\phi}_{\boldsymbol{\xi}}^\top \boldsymbol{\Lambda}_n^{-1}\boldsymbol{\phi}_{\boldsymbol{\delta}} - \boldsymbol{\phi}_{\boldsymbol{\delta}}^\top \boldsymbol{\Lambda}_n^{-1}\boldsymbol{\phi}_{\boldsymbol{\xi}}\right), && (47)\\
&= \frac{2a_n}{2a_n-2}\frac{b_n}{a_n} \lim_{\boldsymbol{\delta}\to\mathbf{0}} \frac{1}{\boldsymbol{\delta}^2}\left((\boldsymbol{\phi}_{\boldsymbol{\delta}} - \boldsymbol{\phi}_{\boldsymbol{\xi}})^\top \boldsymbol{\Lambda}_n^{-1}(\boldsymbol{\phi}_{\boldsymbol{\delta}} - \boldsymbol{\phi}_{\boldsymbol{\xi}})\right), && (48)\\
&= \frac{2a_n}{2a_n-2}\frac{b_n}{a_n}\mathbf{J}_{\boldsymbol{\phi}_{\boldsymbol{\xi}}}^\top \boldsymbol{\Lambda}_n^{-1}\mathbf{J}_{\boldsymbol{\phi}_{\boldsymbol{\xi}}}. && (49)
\end{aligned}$$

Finally, we can assemble our derivative distributions

$$\mathcal{N}(\mathbf{z} \mid \mathbf{J}_{\boldsymbol{\phi}_{\mathbf{x}}}{}^\top \boldsymbol{\mu}_n, \mathbf{J}_{\boldsymbol{\phi}_{\mathbf{x}}}{}^\top \boldsymbol{\Lambda}_n^{-1}\mathbf{J}_{\boldsymbol{\phi}_{\mathbf{x}}}) \quad \text{or} \quad \mathrm{St}(\mathbf{z} \mid \mathbf{J}_{\boldsymbol{\phi}_{\mathbf{x}}}{}^\top \boldsymbol{\mu}_n, \tfrac{b_n}{a_n}\mathbf{J}_{\boldsymbol{\phi}_{\mathbf{x}}}{}^\top \boldsymbol{\Lambda}_n^{-1}\mathbf{J}_{\boldsymbol{\phi}_{\mathbf{x}}}, 2a_n), \tag{50}$$

for the Gaussian and Student-t model respectively.

D Forward Mode Automatic Differentiation

Forward mode automatic differentiation can be represented using dual number algebra. Similar to complex numbers, dual numbers consist of a real part and a dual part, which is a second real number multiplied by a nilpotent ϵ, i.e. $\epsilon^2 = 0$. The real part represents the function value and the dual part represents the directional derivative with respect to the initial input. Both parts can be computed jointly and efficiently by adapting primitive operations.

For example, let $z = 2x + 2\epsilon = \langle 2x, 2\rangle$ be a dual number, where $\mathrm{Re}(z) = 2x$ is the real part and $\mathrm{Du}(z) = 2$ is the dual part. Note that the dual part is the partial derivative of the real part w.r.t. x. Now, let f be the square function $f(x) = x^2$. Applying f to z and cancelling any $\epsilon^2 = 0$ results in another dual number,

$$f(z) = f(\langle 2x, 2\rangle) = (2x+2\epsilon)^2 = 4x^2 + 8x\epsilon + \cancelto{0}{4\epsilon^2} = 4x^2 + 8x\epsilon = \langle 4x^2, 8x\rangle, \tag{51}$$

where $4x^2$ is the real part and $8x$ is the dual part, which is the partial derivative of $4x^2$ w.r.t. x.

For neural network layers, the function value and the directional derivative w.r.t. the input can be derived as closed-form expression, given input value $\mathbf{x}$ and Jacobian $\mathbf{J}$. For example, let $\mathbf{f}$ be an affine transformation with weight matrix $\mathbf{A}$ and bias term $\mathbf{b}$,

$$\mathbf{f}(\langle \mathbf{x}, \mathbf{J}\rangle) = \mathbf{f}(\mathbf{x} + \mathbf{J}\epsilon) = \mathbf{A}(\mathbf{x} + \mathbf{J}\epsilon) + \mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{A}\mathbf{J}\epsilon + \mathbf{b} = \langle \mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{A}\mathbf{J}\rangle. \tag{52}$$

In general,

$$\mathbf{f}(\langle \boldsymbol{\xi}, \mathbf{J} \rangle) = \langle \mathbf{f}(\boldsymbol{\xi}), \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\boldsymbol{\xi}) \mathbf{J} \rangle, \tag{53}$$

which, using dynamic programming, can be turned into efficient implementations.

For a neural network with two hidden layers and element-wise activation functions $\boldsymbol{\sigma}_i$, the output $\mathbf{y}$ given input $\mathbf{x}$ can be expressed using intermediate steps

$$\mathbf{h}_1 = \mathbf{A}_1 \mathbf{x} + \mathbf{b}_1, \qquad \frac{\partial \mathbf{h}_1}{\partial \mathbf{x}} = \mathbf{A}_1, \tag{54}$$

$$\mathbf{z}_1 = \boldsymbol{\sigma}_1(\mathbf{h}_1), \qquad \frac{\partial \mathbf{z}_1}{\partial \mathbf{h}_1} = \frac{\partial \boldsymbol{\sigma}_1}{\partial \mathbf{h}_1}, \tag{55}$$

$$\mathbf{h}_2 = \mathbf{A}_2 \mathbf{z}_1 + \mathbf{b}_2, \qquad \frac{\partial \mathbf{h}_2}{\partial \mathbf{z}_1} = \mathbf{A}_2, \tag{56}$$

$$\mathbf{z}_2 = \boldsymbol{\sigma}_2(\mathbf{h}_2), \qquad \frac{\partial \mathbf{z}_2}{\partial \mathbf{h}_2} = \frac{\partial \boldsymbol{\sigma}_2}{\partial \mathbf{h}_2}, \tag{57}$$

$$\mathbf{y} = \mathbf{A}_3 \mathbf{z}_2 + \mathbf{b}_3, \qquad \frac{\partial \mathbf{y}}{\partial \mathbf{z}_2} = \mathbf{A}_3, \tag{58}$$

where $\mathbf{A}_i$ and $\mathbf{b}_i$ are the weight matrices and bias terms of the corresponding layers, and $\mathbf{h}_i$ and $\mathbf{z}_i$ are intermediate values. The right column lists all intermediate partial derivatives.

Using the dual number notation and the general relationship from Equation (53), and initializing the Jacobian with the identity matrix $\mathbf{I}$, we can write

$$\mathbf{h}_1 = \langle \mathbf{A}_1 \mathbf{x} + \mathbf{b}_1, \quad \mathbf{A}_1 \rangle, \tag{59}$$

$$\mathbf{z}_1 = \langle \boldsymbol{\sigma}_1(\mathbf{h}_1), \quad \frac{\partial \boldsymbol{\sigma}_1}{\partial \mathbf{h}}(\mathbf{h}_1)\mathbf{A}_1 \rangle, \tag{60}$$

$$\mathbf{h}_2 = \langle \mathbf{A}_2 \mathbf{z}_1 + \mathbf{b}_2, \quad \mathbf{A}_2 \frac{\partial \boldsymbol{\sigma}_1}{\partial \mathbf{h}}(\mathbf{h}_1)\mathbf{A}_1 \rangle, \tag{61}$$

$$\mathbf{z}_2 = \langle \boldsymbol{\sigma}_2(\mathbf{h}_2), \quad \frac{\partial \boldsymbol{\sigma}_2}{\partial \mathbf{h}}(\mathbf{h}_2)\mathbf{A}_2 \frac{\partial \boldsymbol{\sigma}_1}{\partial \mathbf{h}}(\mathbf{h}_1)\mathbf{A}_1 \rangle, \tag{62}$$

$$\mathbf{y} = \langle \mathbf{A}_3 \mathbf{z}_2 + \mathbf{b}_3, \quad \mathbf{A}_3 \frac{\partial \boldsymbol{\sigma}_2}{\partial \mathbf{h}}(\mathbf{h}_2)\mathbf{A}_2 \frac{\partial \boldsymbol{\sigma}_1}{\partial \mathbf{h}}(\mathbf{h}_1)\mathbf{A}_1 \rangle, \tag{63}$$

where $(\partial \boldsymbol{\sigma}_i / \partial \mathbf{h})(\mathbf{h}_i)$ is the partial derivative of the activation function $\boldsymbol{\sigma}_i$ w.r.t. its input and evaluated at $\mathbf{h}_i$. In particular, assuming that closed-form expressions are available for $\partial \boldsymbol{\sigma}_i / \partial \mathbf{h}$, the dual part of each intermediate result only depends on previously computed values. Thus, dynamic programming can be leveraged to jointly compute the real and the dual part with a single forward pass.

To confirm that the dual part is indeed the desired partial derivative of output $\mathbf{y}$ w.r.t. initial input $\mathbf{x}$, we apply the chain rule of derivatives, such that $\partial \mathbf{y} / \partial \mathbf{x}$ factorizes as

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}}{\partial \mathbf{z}_2} \frac{\partial \mathbf{z}_2}{\partial \mathbf{h}_2} \frac{\partial \mathbf{h}_2}{\partial \mathbf{z}_1} \frac{\partial \mathbf{z}_1}{\partial \mathbf{h}_1} \frac{\partial \mathbf{h}_1}{\partial \mathbf{x}}. \tag{64}$$

Substituting the corresponding expressions confirms the equivalence.

E Implementation Details

For this work, we used the `PyTorch` library [56]. Additionally, for the MFVI baselines, we used `Pyro` [5]. All models are implemented to support multivariate inputs and multiple outputs.

For all BLL models, we set $\boldsymbol{\mu}_0 = \mathbf{0}$ and $\boldsymbol{\Lambda}_0$ to a diagonal matrix with m ($+k$ if using identity features, $+1$ if using bias term) distinct parameters along the diagonal. Identity features and bias term for the Bayesian weights $\boldsymbol{\beta}$ were always used. Hidden bias terms were enabled for all neural network layers. We conducted experiments with tanh and leaky relu activation functions, the number of hidden neural network units varied, they are listed in Section I. The prior weight precision matrix $\boldsymbol{\Lambda}_0$ is initialized to identity.

For the Gaussian model, we use a diagonal covariance matrix parameter $\mathbf{\Sigma}$ to represent the aleatoric uncertainty independently for each output dimension. We initialized $\mathbf{\Sigma}$ to identity.

For the Student-t model, we use a degree of freedom parameter ν_0 and a diagonal, positive definite matrix $\mathbf{V}_0$ to represent an inverse-Wishart distribution instead. We initialized ν_0 and all diagonal entries of $\mathbf{V}_0$ to 2, to match initial values of 1 for the inverse-gamma distribution in the one-dimensional output case.

If prior parameter values are estimated jointly via backpropagation of the marginal likelihood (and fKL) objective they are learned in log-space using a proxy variable to assert positive value constraints.

For the noise distribution used to create index sets, we set $\gamma = 0.01$, such that the standard deviaton for each input dimension equals 0.1 in whitened space.

For MFVI, we used independent $\mathcal{N}(0, \omega/\sqrt{n_{\text{in}}})$ priors for all neural network weights, where n_{in} is the number of input features to the corresponding layer and $\omega = 4$, such that the GP limit exists [50]. Bias terms with independent $\mathcal{N}(0, 1)$ priors were enabled for all layers. The variational distribution was implemented using `AutoDiagonalNormal` from `Pyro`. For optimization, we used `SVI` and `Trace_ELBO` from `Pyro` for optimization. To compute predictions for validation or evaluation, we drew 100 samples from the weight distributions.

Further details for individual experiments are listed in Section I.

Computational Resources The experiments were conducted on a computer with an AMD Ryzen 9 3900X 12-Core Processor, an Nvidia RTX 2080 graphics card and 64GB of RAM. Large models and datasets were run on an Nvidia DGX workstation.

Algorithm 1 Latent derivative Bayesian last layer training.

```
1: \\ Initialize prior and neural network parameters
2: $\boldsymbol{\psi}_0 := \mathbf{\Lambda}_0, \mathbf{\Sigma} \leftarrow \mathbf{I}, \mathbf{I}$ (Gaussian)   or   $\boldsymbol{\psi}_0 := \mathbf{\Lambda}_0, \nu_0, \mathbf{V}_0 \leftarrow \mathbf{I}, 2, 2\mathbf{I}$ (Student-$t$)
3: $\boldsymbol{\theta} \sim \mathcal{N}$
4: for num_epochs do
5:     \\ Compute features, Bayesian update, marginal likelihood
6:     $\Phi \leftarrow \boldsymbol{\phi}(\mathbf{X}; \boldsymbol{\theta})$
7:     $\boldsymbol{\psi}_n \leftarrow \text{Bayes}(\Phi, \mathbf{y}, \boldsymbol{\psi}_0)$            ▷ Eq. (15) & (21)
8:     $\mathcal{L} \leftarrow -\frac{1}{n}\text{LLH}(\boldsymbol{\psi}_0, \boldsymbol{\psi}_n, n)$            ▷ Eq. (18) & (26)
9:     \\ Draw samples, compute Jacobian and fKL using posterior
10:    $\mathbf{S} \sim \mathcal{N}(\mathbf{X}, \gamma\mathbf{I})$
11:    $\mathbf{J}_{\boldsymbol{\phi}} \leftarrow \boldsymbol{\phi}(\mathbf{S}; \boldsymbol{\theta})$
12:    $\mathcal{L} \leftarrow \frac{1}{n}\text{fKL}(\mathbf{J}_{\boldsymbol{\phi}}, \boldsymbol{\psi}_n, \sigma_{\mathbf{z}}^2)$            ▷ Eq. (10)
13:    \\ Compute gradients, update neural network and (optionally) prior parameters
14:    $\boldsymbol{\theta}, \boldsymbol{\psi}_0 \leftarrow \text{Adam}(\mathcal{L})$
15: end for
```

F Computational Complexity

In terms of prediction, the BLL and the LDBLL perform the same operations, given that the LD setting only affects the training of the feature space. Therefore, they also share the same computational complexity, namely $\mathcal{O}(m^2)$ time per prediction, where the feature space dimension m is assumed to be the largest hidden layer dimension and the computation of activation functions is neglected. The computational complexity of the training procedure differs for the BLL and the LDBLL since, in addition to the conventional marginal likelihood objective which is used for the BLL, the LDBLL also requires evaluation of the fKL objective. With the same assumptions about the hidden layer dimensions and computation of activation functions, the marginal likelihood objective can be computed in $\mathcal{O}(nm^2{+}m^3)$ time, where $n = |\mathcal{D}|$ is the size of the training set. The additional fKL objective which is required for the LDBLL can be computed in $\mathcal{O}(|\mathcal{T}|(mk^2{+}k^3))$ time, where $|\mathcal{T}|$ is the number of points in the index set and k is the number of input dimensions of the latent function f. A complete training epoch for the LDBLL can thus be computed in $\mathcal{O}(nm^2{+}m^3 + |\mathcal{T}|(mk^2{+}k^3))$ time. Setting the index set $\mathcal{T}$ to Gaussian samples near the training data implies $|\mathcal{T}| = |\mathcal{D}| = n$, factorizing the computational complexity for a single training epoch into $\mathcal{O}(n(m^2{+}mk^2{+}k^3){+}m^3)$. We see that for $k \leq m$, the additional complexity introduced by the fKL objective is manageable. However, the cubic scaling of the complexity w.r.t. k highlights the need for approximate techniques for very high-dimensional inputs such as images. Although not evident in the asymptotic analysis of computational complexity, if $\mathcal{T} \neq \mathcal{D}$ then a single training epoch requires two separate forwarded passes through $\boldsymbol{\phi}$, to compute the features for $\mathcal{D}$ and the features and their Jacobians for $\mathcal{T}$ respectively. If $\mathcal{T} = \mathcal{D}$ then the same features can be used for both objectives and their Jacobians can be computed jointly in the same forward pass which saves one forward pass of feature computations. However, in practice, the difference is negligible because the feature computations are rather insignificant compared to the computation and backpropagation of Jacobians.

G M- vs. I-Projection for the functional KL

As the covariance of $\mathbf{z}$ contains an inner product of the Jacobians ($\mathbf{J}_{\boldsymbol{\phi}_{\mathbf{x}}}{}^{\top}\boldsymbol{\Lambda}_n^{-1}\mathbf{J}_{\boldsymbol{\phi}_{\mathbf{x}}}$), it is highly structured in a way that approximates the Hessian (i.e. as in Gauss-Newton optimization). Therefore the structure of the covariance depends strongly on $\mathbf{x}$, the BLL and the underlying function being modelled. Rather than design π to reflect this variation, ideally the objective would be less concerned with the specific *structure* and rather the *size* of the covariance. Fortunately, for the KL divergence between multivariate Gaussian distributions, the M-projection enforces the small covariance penalty with a trace term and the large covariance penalty via the entropy difference. As the trace and entropy terms can be viewed to act on the covariance's structure and size respectively, for the 'max entropy' latent derivative objective, the M-projection encourages $\mathbf{z}$ to grow in entropy. It was observed empirically that the M-projection was indeed better than the I-projection for training the model.

H Ablation and Hyperparameter Sensitivity Study

To illustrate the importance and sensitivity of the LDBLL's model parameters, we provide some visualizations of their effects. We use the function of Figure 2 as a reference with $[50, 50]$ tanh networks trained to convergence, unless specified otherwise.

Network Size

As Bayesian models should scale gracefully with model complexity [62], in Figure 7 we show that the model predictions are comparable with significantly increasing network size.

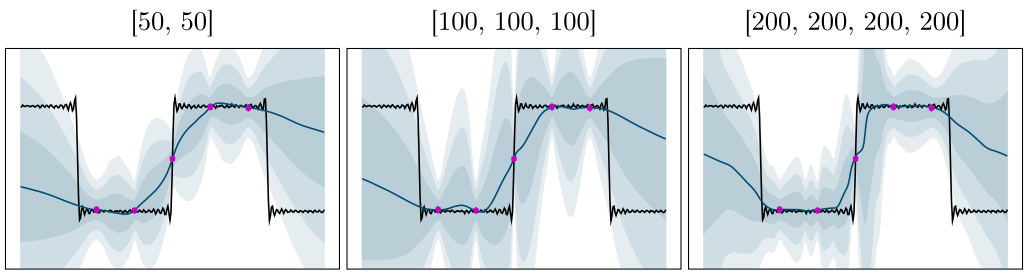

Figure 7: LDTBLL with increasing network size. While the fit does vary with the model size, this is mainly due to the increased fidelity. The mean and variance maintain a reasonably consistent shape throughout.

Activation Function

In Figure 8 we show how the hypothesis space changes with activation function.

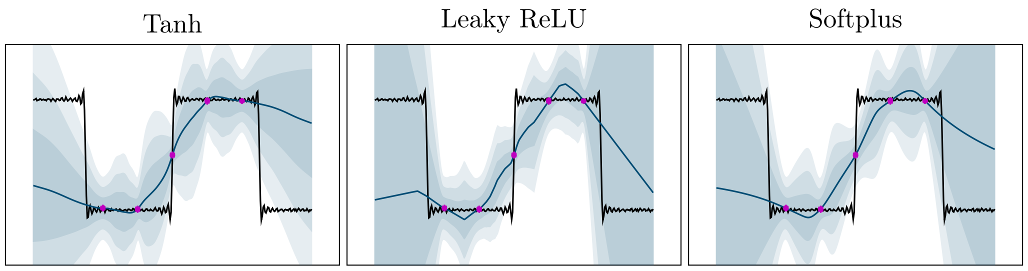

Figure 8: LDTBLL with increasing a range of activation functions. Although the hypothesis space changes (i.e. smoothness), the mean and variance remain consistent.

Latent Derivative Variance

In Figure 9, we demonstrate that reducing $\mathbf{\Sigma}_\pi$ reduces the diversity of the feature space, but for increasing $\mathbf{\Sigma}_\pi$ a reduced effect is seen due to the limited network capacity.

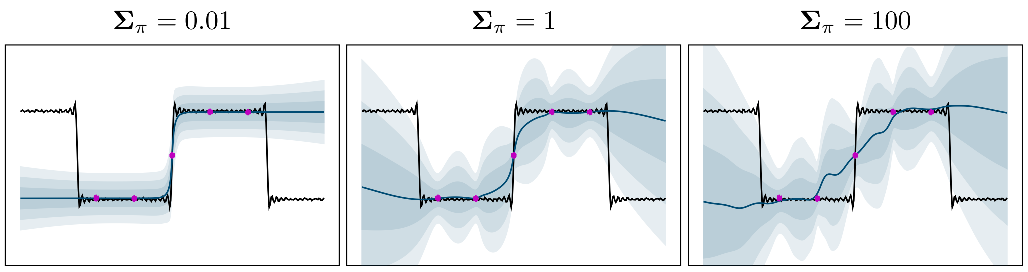

Figure 9: LDTBLL with varying latent derivative variance. The prior controls the variety of the function space, but this is mainly only an issue for small values of $\mathbf{\Sigma}_\pi$.

I Experiments

In this section, we explain our experiments and display hyperparameters and detailed numerical results.

I.1 Nonlinear Regression

We compared the BLL and LDBLL across a set of popular BNN baselines: MFVI, Monte Carlo dropout (MC dropout), ensembles and SWAG, along with a Gaussian process and MAP (maximum likelihood neural network training with weight decay) baseline. We evaluated these models for nonlinear regression experiments on 'gap' tasks, namely Cartpole, CO2, Sarcos and WAM, and 'standard' UCI benchmarks.

For all experiments, we used Adam [35] with default parameter configurations except the learning rate. All learning rates were handtuned for every pair of model and data. The number of training epochs were selected using a validation set that consists of 20% of the training data. We implemented early stopping by tracking the validation log-likelihood up until a maximum number of epochs. The number of training epochs that yielded the highest validation log-likelihood is used to re-train on the full training data. For BLL and LDBLL, we compute the validation log-likelihood after every epoch, whereas for implicit predictive distributions, due to the necessity of sampling during prediction, it is too expensive to compute the validation log-likelihood after every epoch. Instead, we updated the validation log-likelihood every 100 epochs. Since tracking the validation log-likelihood is also too expensive for GP regression, we stopped optimization when the average marginal likelihood of the past κ epochs decreased less than threshold ρ compared to the average of the previous κ epochs. For all datasets, κ was set to 11 and ρ was set to 1e-4, except UCI Naval where ρ was set to 1e-2. The number of hidden units, learning rates and maximum number of epochs are listed in Table 6 for 'gap' and Table 17 for 'standard' tasks, respectively. With respect to model-specific hyperparameters of the baselines, we either adopted the recommended values, or manually chose a reasonable value that performed well across tasks. This was to ensure a fair comparison to the BLL, which also had fixed hyperparameters across tasks.

For BLL and LDBLL, hyperparameters, such as the weight prior and aleatoric noise, were optimized via backpropagation using the marginal likelihood. For the LDBLL fKL objective, the model observation noise was used instead of a fixed noise prior. The index set was created by adding noise to the training data, except for CO2, where the training data itself was used as index set because adding noise caused problems with the sinosoidal features. For MFVI, ensembles, MC dropout, SWAG and MAP, a minibatch size of 32 was used for all regression experiments. For the ensembles, 5 models were used. For MC dropout, a dropout probability of 0.2 was used. For SWAG, the sampling learning rate was set to double the training learning rate, and 30 steps were used when sampling. As an additional note: the baselines above are sometimes trained with an additional network to model heteroskedastic noise. We do not do this and assume Gaussian homoskedastic noise across all models.

CO2
The Mauna Loa atmospheric carbon dioxide dataset contains CO2 measurements over several decades. To encode the periodicity, we added $\sin(2\pi\mathbf{x})$ and $\cos(2\pi\mathbf{x})$ features, such that the input was three-dimensional.

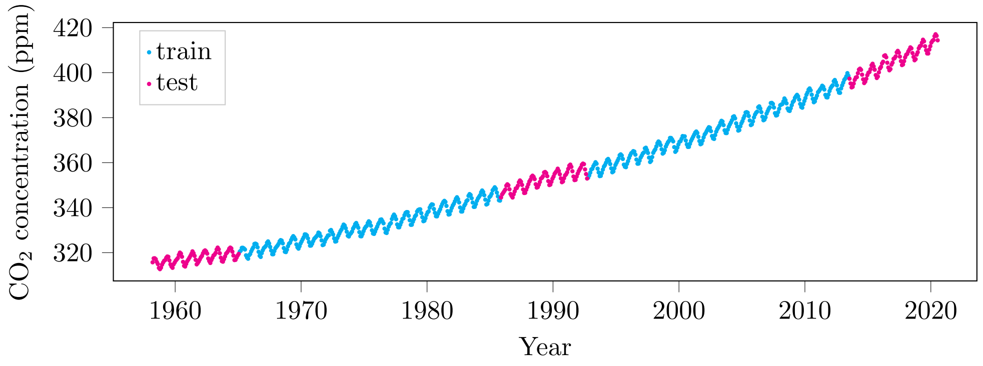

Figure 10: CO2 dataset with test gap regions.

Table 2: Regression results for the co2 dataset, means and standard errors over 10 seeds

CO2		TRAIN			TEST		
$n = 498,\ k = 3$		RMSE ↓	LLH ↑	ENTR ↓	RMSE ↓	LLH ↑	ENTR ↑
GP	RBF	0.52 ± 0.00	-0.77 ± 0.00	0.82 ± 0.00	1.70 ± 0.00	-4.49 ± 0.00	0.87 ± 0.00
GBLL	LRELU	0.37 ± 0.01	-0.52 ± 0.03	0.75 ± 0.04	2.53 ± 0.26	-11.23 ± 1.95	0.79 ± 0.04
	TANH	0.50 ± 0.01	-0.74 ± 0.02	0.88 ± 0.02	2.59 ± 0.17	-8.44 ± 0.92	0.93 ± 0.02
LDGBLL	LRELU	0.46 ± 0.01	-1.50 ± 0.03	1.97 ± 0.03	2.59 ± 0.71	-2.04 ± 0.03	2.14 ± 0.03
	TANH	0.42 ± 0.00	-1.18 ± 0.02	1.62 ± 0.03	2.38 ± 0.14	-2.52 ± 0.16	1.79 ± 0.06
MFVI	LRELU	0.39 ± 0.01	-0.50 ± 0.02	0.62 ± 0.04	1.82 ± 0.07	-7.23 ± 0.59	0.67 ± 0.05
	TANH	0.40 ± 0.00	-0.53 ± 0.01	0.66 ± 0.03	3.35 ± 0.11	-26.90 ± 1.08	0.65 ± 0.03
ENSEMBLE	LRELU	0.35 ± 0.01	-0.40 ± 0.01	0.59 ± 0.01	2.10 ± 0.03	-6.67 ± 0.34	0.77 ± 0.01
	TANH	0.41 ± 0.00	-0.55 ± 0.00	0.69 ± 0.01	2.58 ± 0.03	-9.84 ± 0.41	0.79 ± 0.02
DROPOUT	LRELU	0.59 ± 0.03	-2.07 ± 0.00	2.55 ± 0.00	2.18 ± 0.09	-2.42 ± 0.01	2.78 ± 0.00
	TANH	0.91 ± 0.00	-2.15 ± 0.00	2.61 ± 0.00	5.19 ± 0.03	-2.96 ± 0.01	2.71 ± 0.00
SWAG	LRELU	7.16 ± 0.56	-3.27 ± 0.07	3.38 ± 0.08	10.73 ± 1.08	-3.56 ± 0.11	3.64 ± 0.08
MAP	LRELU	0.34 ± 0.00	-0.33 ± 0.01	0.35 ± 0.01	1.93 ± 0.03	-15.73 ± 0.50	0.35 ± 0.01
	TANH	0.40 ± 0.00	-0.52 ± 0.00	0.53 ± 0.00	2.01 ± 0.03	-12.09 ± 0.33	0.53 ± 0.00

Cartpole
Telemetry is recorded from a Quanser cartpole system performing a swing-up maneuver. We use the dynamic state (position, velocity and acceleration) of the cart and pole for inverse dynamics modeling of the drive torque.

Table 3: Regression results for the cartpole dataset, means and standard errors over 10 seeds

CARTPOLE		TRAIN			TEST		
$n = 665,\ k = 6$		RMSE ↓	LLH ↑	ENTR ↓	RMSE ↓	LLH ↑	ENTR ↑
GP	RBF	0.87 ± 0.00	-1.45 ± 0.00	1.68 ± 0.00	13.64 ± 0.00	-4.01 ± 0.00	4.18 ± 0.00
GBLL	LRELU	0.26 ± 0.02	-1.26 ± 0.10	1.74 ± 0.10	221.60 ± 55.83	-115.94 ± 50.48	3.94 ± 0.14
	TANH	0.11 ± 0.01	-0.76 ± 0.25	1.25 ± 0.25	9.47 ± 0.93	-27.95 ± 9.96	2.32 ± 0.26
LDGBLL	LRELU	0.25 ± 0.02	-0.94 ± 0.16	1.40 ± 0.17	179.20 ± 79.00	-11.68 ± 2.14	4.85 ± 0.29
	TANH	0.17 ± 0.02	-0.71 ± 0.26	1.18 ± 0.27	10.32 ± 1.98	-8.07 ± 1.60	3.19 ± 0.25
MFVI	LRELU	0.41 ± 0.10	-0.37 ± 0.22	0.76 ± 0.22	10.69 ± 2.13	-12.19 ± 3.08	3.13 ± 0.18
	TANH	0.32 ± 0.12	0.28 ± 0.39	0.07 ± 0.42	7.72 ± 0.55	-650.53 ± 358.66	1.27 ± 0.32
ENSEMBLE	LRELU	0.19 ± 0.05	0.35 ± 0.30	0.09 ± 0.30	37.03 ± 4.88	-5.20 ± 0.11	5.39 ± 0.14
	TANH	0.56 ± 0.06	-0.78 ± 0.13	1.00 ± 0.14	5.50 ± 1.22	-3.75 ± 0.28	3.06 ± 0.14
DROPOUT	LRELU	0.35 ± 0.01	-1.26 ± 0.01	1.73 ± 0.01	4.59 ± 0.22	-3.73 ± 0.14	2.48 ± 0.04
	TANH	0.70 ± 0.01	-1.40 ± 0.01	1.80 ± 0.01	10.96 ± 0.35	-27.84 ± 1.54	1.84 ± 0.01
SWAG	LRELU	1.21 ± 0.09	-1.51 ± 0.09	1.63 ± 0.06	49.39 ± 8.45	-106.72 ± 34.69	3.06 ± 0.12
MAP	LRELU	0.38 ± 0.02	-0.74 ± 0.08	1.09 ± 0.09	52.50 ± 7.62	-5800.91 ± 2276.39	1.09 ± 0.09
	TANH	0.60 ± 0.03	-0.95 ± 0.06	1.12 ± 0.08	6.49 ± 0.62	-64.36 ± 21.45	1.12 ± 0.08

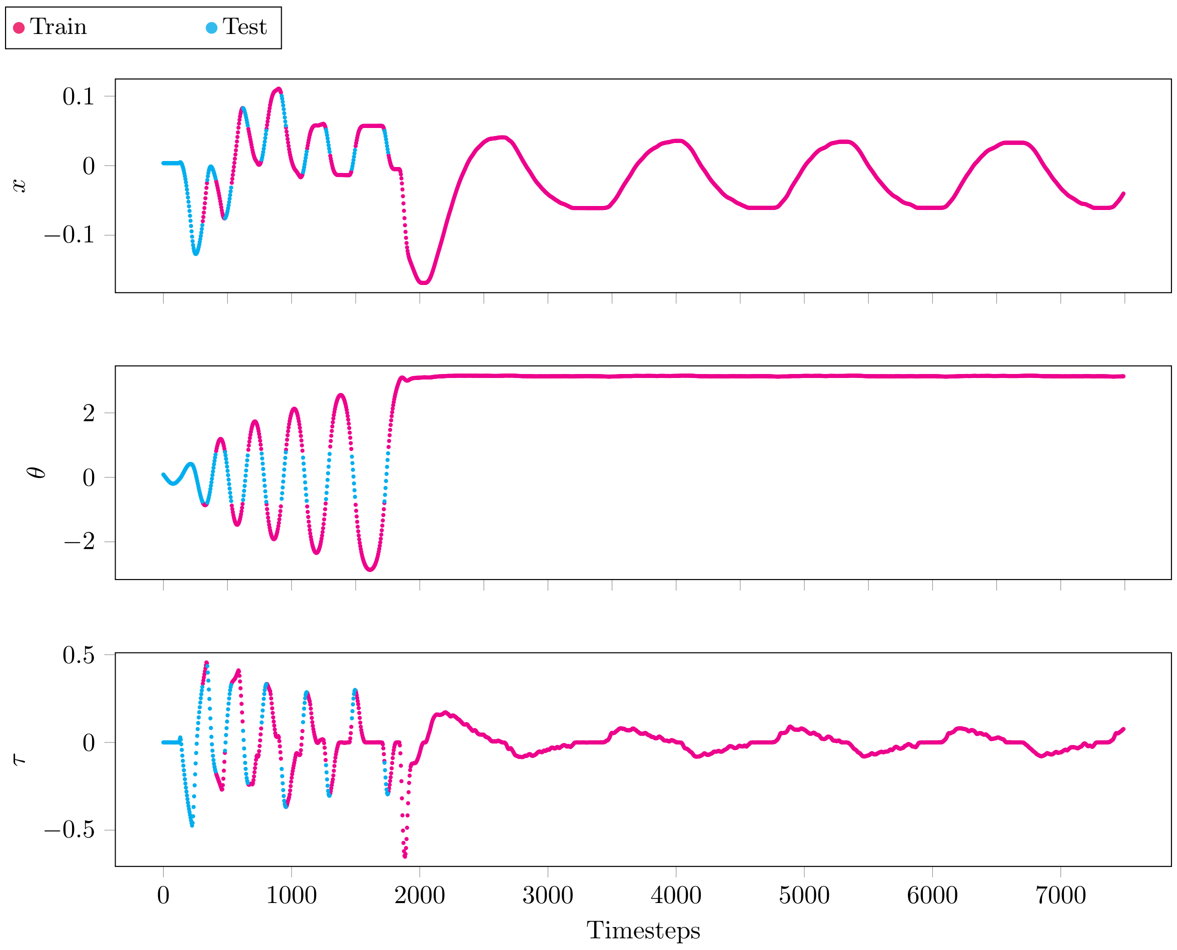

Figure 11: Visualization of the Quanser cartpole swing-up dataset, depicting cart position x, pole angle θ and cart drive torque τ. The complete dataset also includes the velocities and accelerations. Note that the first ~2000 sample contains the swing-up, while the subsequent telemetry is the sustained stabilization. As a consequence, the data distribution is significantly non-uniformly distributed in the state space, so uncertainty-driven active learning is superior to a random data selection strategy. This figure illustrates the gap split, where the $\theta < 45°$ region is used for testing. For active learning the whole dataset is accessible and a different test split is used.

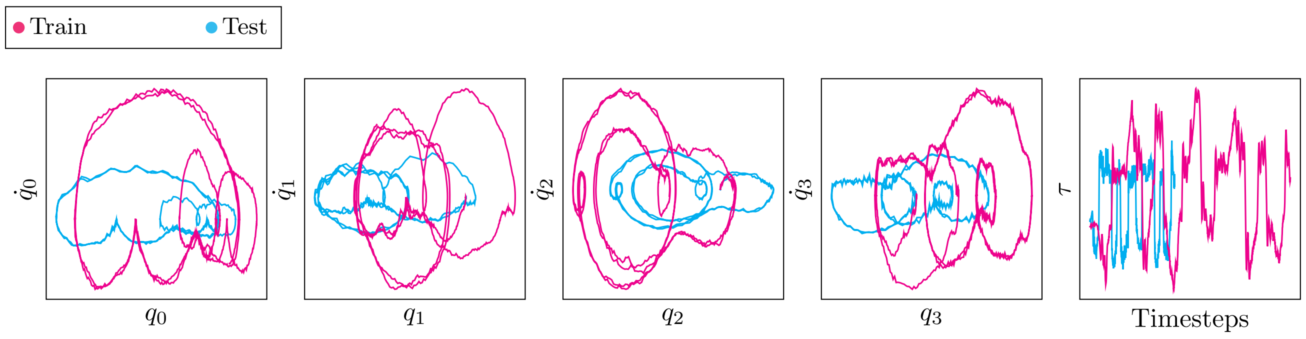

Figure 12: Phase plots for each join state q and base drive torque τ to illustrate the gap dataset generated by running the same trajectory at two different speeds on the Barret WAM 4 DOF manipulator.

Sarcos
The Sarcos dataset [79] is the telemetry collected from a 7-DOF robot manipulator. The dataset is typically used for modelling the inverse dynamics, mapping the dynamic state (position, velocities and accelerations) to the torques supplied to the electric drives of each joint. To test epistemic uncertainty, we designed a new split of this dataset to test OOD prediction, where the 5th joint moving $< 10°$ induces a significant bias in the torque of the 6th joint due to gravity (Figure 13). Therefore the test data for this split contains values not present in both the inputs and targets of the training data.

Table 4: Regression results for the sarcos dataset, means and standard errors over 10 seeds

SARCOS		TRAIN			TEST		
$n = 19172,\ k = 21$		RMSE ↓	LLH ↑	ENTR ↓	RMSE ↓	LLH ↑	ENTR ↑
GP	RBF	0.07 ± 0.00	0.84 ± 0.01	-0.46 ± 0.01	2.75 ± 0.00	-5.07 ± 0.03	1.35 ± 0.00
GBLL	LRELU	0.06 ± 0.01	1.04 ± 0.08	-0.67 ± 0.05	3.69 ± 0.15	-379.72 ± 53.31	-0.55 ± 0.05
	TANH	0.09 ± 0.01	0.90 ± 0.05	-0.71 ± 0.01	4.08 ± 0.15	-403.15 ± 30.66	-0.52 ± 0.01
LDGBLL	LRELU	0.09 ± 0.00	0.66 ± 0.04	-0.28 ± 0.04	2.80 ± 0.11	-51.98 ± 6.59	0.09 ± 0.03
	TANH	0.05 ± 0.00	1.18 ± 0.01	-0.78 ± 0.01	2.51 ± 0.03	-169.77 ± 5.08	-0.59 ± 0.01
MFVI	LRELU	0.08 ± 0.00	1.05 ± 0.03	-0.87 ± 0.02	2.95 ± 0.19	-52.23 ± 5.72	0.12 ± 0.04
	TANH	0.07 ± 0.00	1.04 ± 0.01	-0.73 ± 0.01	2.13 ± 0.05	-59.30 ± 4.36	-0.19 ± 0.02
ENSEMBLE	LRELU	0.05 ± 0.00	1.50 ± 0.02	-1.25 ± 0.02	3.01 ± 0.05	-7.64 ± 0.84	1.43 ± 0.03
	TANH	0.06 ± 0.00	1.37 ± 0.01	-1.17 ± 0.01	2.30 ± 0.02	-13.24 ± 0.83	0.77 ± 0.02
DROPOUT	LRELU	0.08 ± 0.00	0.73 ± 0.00	-0.33 ± 0.00	2.67 ± 0.04	-8.58 ± 0.50	0.92 ± 0.02
	TANH	0.13 ± 0.00	0.31 ± 0.00	0.07 ± 0.00	2.08 ± 0.02	-25.92 ± 0.62	0.19 ± 0.00
SWAG	LRELU	0.10 ± 0.00	0.83 ± 0.02	-0.60 ± 0.03	3.03 ± 0.07	-15.34 ± 0.47	0.79 ± 0.03
MAP	LRELU	0.04 ± 0.00	0.81 ± 0.00	-0.34 ± 0.00	3.27 ± 0.13	-199.49 ± 15.53	-0.34 ± 0.00
	TANH	0.06 ± 0.00	0.78 ± 0.00	-0.33 ± 0.00	2.67 ± 0.12	-121.14 ± 10.30	-0.33 ± 0.00

WAM
This dataset is also derived from a robotic manipulator, the cable-driven 4 DOF Barrett WAM. However, here the distribution shift is generated by demanding the same complex motion at different velocities. By training on a slower motion and evaluating the inverse dynamics model for data collected at a faster speed, the prediction considers the same trajectory but now with higher variance in the values of the state and input due to the larger accelerations at play.

Table 5: Regression results for the wam dataset, means and standard errors over 10 seeds

WAM		TRAIN			TEST		
$n = 16497,\ k = 12$		RMSE ↓	LLH ↑	ENTR ↓	RMSE ↓	LLH ↑	ENTR ↑
GP	RBF	0.11 ± 0.00	0.23 ± 0.01	0.20 ± 0.01	1.63 ± 0.01	-2.10 ± 0.01	1.54 ± 0.00
GBLL	LRELU	0.08 ± 0.01	1.07 ± 0.10	-1.11 ± 0.05	2.18 ± 0.06	-378.90 ± 41.63	-1.08 ± 0.05
	TANH	0.12 ± 0.00	0.68 ± 0.03	-0.65 ± 0.03	3.26 ± 0.12	-173.61 ± 11.61	-0.37 ± 0.02
LDGBLL	LRELU	0.16 ± 0.01	0.40 ± 0.04	-0.23 ± 0.05	1.87 ± 0.06	-35.36 ± 4.17	-0.05 ± 0.04
	TANH	0.12 ± 0.00	0.65 ± 0.03	-0.47 ± 0.03	3.12 ± 0.07	-106.86 ± 8.28	-0.18 ± 0.03
MFVI	LRELU	0.05 ± 0.00	1.62 ± 0.02	-1.53 ± 0.02	3.36 ± 0.45	-315.55 ± 26.33	-0.70 ± 0.08
	TANH	0.05 ± 0.00	1.64 ± 0.02	-1.51 ± 0.02	1.46 ± 0.02	-311.69 ± 19.86	-1.43 ± 0.02
ENSEMBLE	LRELU	0.03 ± 0.00	1.91 ± 0.00	-1.57 ± 0.00	1.73 ± 0.03	-4.79 ± 0.26	1.26 ± 0.06
	TANH	0.03 ± 0.00	1.95 ± 0.01	-1.58 ± 0.00	1.36 ± 0.00	-17.73 ± 0.88	0.06 ± 0.01
DROPOUT	LRELU	0.06 ± 0.00	0.81 ± 0.00	-0.38 ± 0.00	1.41 ± 0.02	-15.46 ± 0.40	0.17 ± 0.01
	TANH	0.10 ± 0.00	0.56 ± 0.00	-0.17 ± 0.00	1.29 ± 0.00	-18.28 ± 0.12	-0.10 ± 0.00
SWAG	LRELU	0.08 ± 0.00	1.04 ± 0.05	-0.78 ± 0.06	1.66 ± 0.03	-29.49 ± 2.50	-0.08 ± 0.05
MAP	LRELU	0.04 ± 0.00	0.51 ± 0.00	-0.02 ± 0.00	2.04 ± 0.05	-39.54 ± 2.00	-0.02 ± 0.00
	TANH	0.05 ± 0.00	0.50 ± 0.00	-0.02 ± 0.00	1.73 ± 0.02	-26.92 ± 0.66	-0.02 ± 0.00

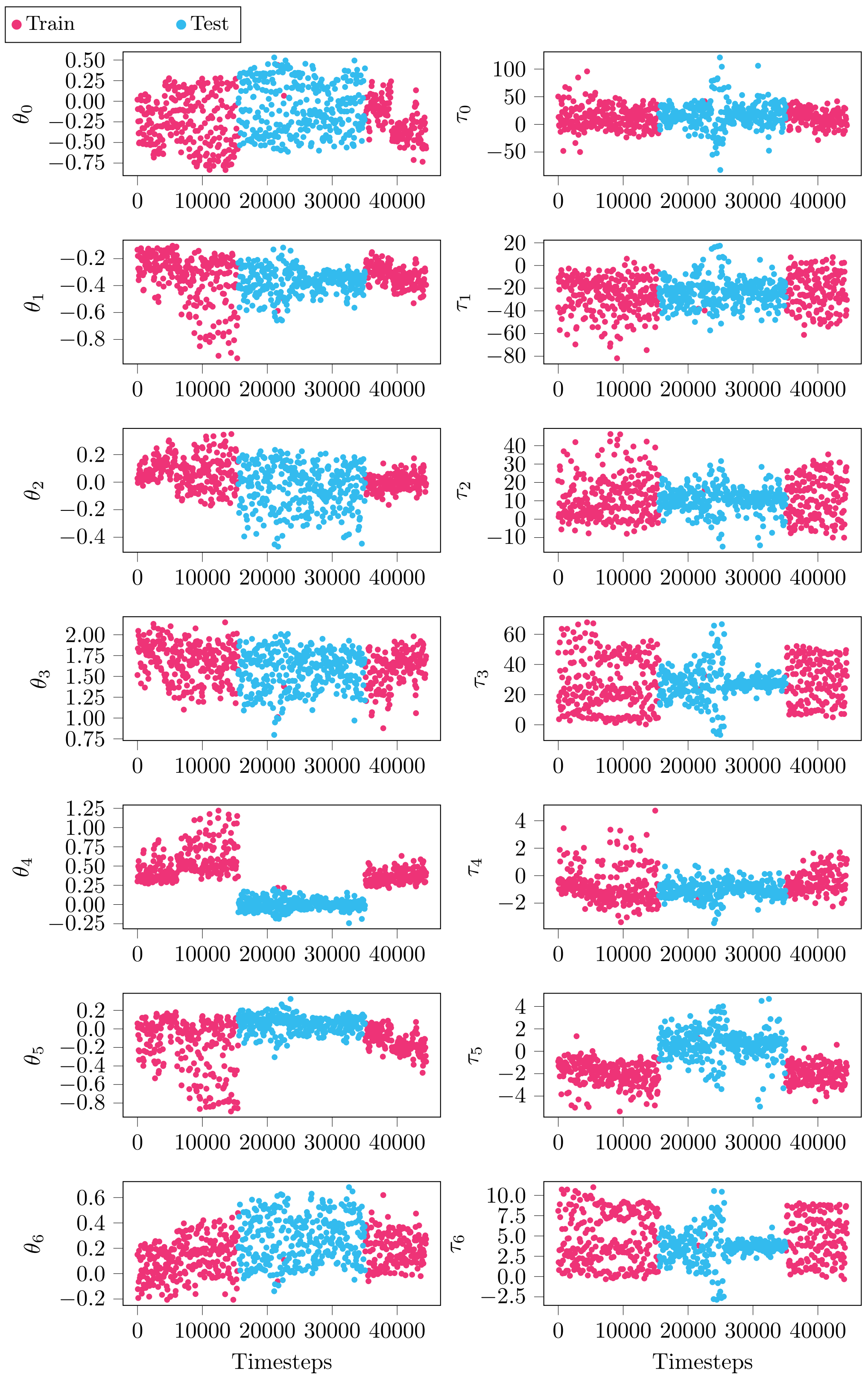

Figure 13: Visualization of the Sarcos data, with joint positions θ and drive torques τ. The gap (●) is generated from θ_4 during the period of sustained displacement, and the regression target is τ_5 due to the induced offset. Note that the data has been downsampled by a factor of 60 for plotting, so the high frequency component of the data are not visible.

Table 6: Number of hidden units, learning rates and maximum number of epochs for gap regression tasks.

GAP		CO2	CARTPOLE	SARCOS	WAM
HIDDEN DIMS		50 50	50 50	200 200	200 200
GBLL	LRELU	1E-3, 10000	1E-3, 7000	1E-3, 8000	5E-4, 15000
	TANH	1E-3, 10000	1E-3, 7000	1E-3, 7000	5E-4, 15000
LDGBLL	LRELU	1E-3, 10000	1E-3, 10000	1E-3, 10000	5E-4, 15000
	TANH	1E-3, 10000	1E-3, 10000	1E-3, 10000	5E-4, 15000
MFVI	LRELU	1E-3, 50000	1E-3, 50000	1E-3, 40000	1E-3, 60000
	TANH	1E-3, 50000	1E-3, 50000	1E-3, 40000	1E-3, 60000
DROPOUT	LRELU	1E-3, 50000	1E-3, 50000	1E-3, 40000	1E-3, 60000
	TANH	1E-3, 50000	1E-3, 50000	1E-3, 40000	1E-3, 60000
ENSEMBLE	LRELU	1E-3, 1000	1E-3, 8000	1E-3, 40000	1E-3, 3000
	TANH	1E-3, 1000	1E-3, 10000	1E-3, 3000	1E-3, 3000
DROPOUT	LRELU	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000
	TANH	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000
SWAG	LRELU	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000
MAP	LRELU	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000
	TANH	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000
GP	RBF	1E-2, 2000	1E-2, 1000	1E-2, 5000	1E-3, 5000

Criticism of the 'In-between uncertainty' gap experiment and benchmarks
The gap splits of the UCI benchmark [23] were motivated to evaluate the epistemic uncertainty of Bayesian Neural Networks, which is typically not necessary in standard regression tasks where the train and test data are drawn from the same distribution. The introduction of this benchmark will hopefully lead to greater scrutiny of the quality of Bayesian neural networks as Bayesian statistical models.

However, evaluating the benchmark, the authors discovered several weaknesses in the experiment's initial formulation that hinders its utility as a useful benchmark. The gap splits are generated by creating k splits for a k dimensional input, and each split contains a 'gap' test set defined by the corresponding dimension (i.e. the second split has a gap in the 2nd input). The gap / test indices are computed by sorting the data along the gap dimension, and extracting the central third. Due to the definition of the splits, they do not represent a *statistical* effect, but a *structural* one. Performance between splits depends heavily on the relevance of the input to the regression problem, therefore the standard error in performance is influenced, potentially dominated, by the splits themselves. Also, there is no guarantee that the gap exhibits 'interesting behavior', e.g. OOD data. If the gap is approximately linear, then crude, overconfident predictions could achieve deceptively good results. Due to these reasons, we chose to curate our own gap datasets that we hope is adopted as a standard benchmark.

UCI

In this subsection, we display all regression results for the 'standard' UCI benchmarks. We also report results on standard regression for the sarcos dataset.

Table 7: Regression results for the boston dataset, means and standard errors over 20 seeds

BOSTON		TRAIN			TEST		
$n = 455,\ k = 13$		RMSE ↓	LLH ↑	ENTR ↓	RMSE ↓	LLH ↑	ENTR ↓
GP	RBF	1.23 ± 0.01	-1.86 ± 0.01	2.19 ± 0.01	2.83 ± 0.16	-2.41 ± 0.06	2.39 ± 0.01
GBLL	LRELU	0.23 ± 0.03	-2.35 ± 0.04	2.85 ± 0.04	4.19 ± 0.17	-2.90 ± 0.05	2.85 ± 0.04
	TANH	0.56 ± 0.06	-2.74 ± 0.04	3.24 ± 0.04	4.61 ± 0.23	-3.06 ± 0.03	3.24 ± 0.04
LDGBLL	LRELU	0.52 ± 0.02	-2.05 ± 0.03	2.53 ± 0.03	3.38 ± 0.18	-2.60 ± 0.04	2.61 ± 0.03
	TANH	0.65 ± 0.02	-2.06 ± 0.03	2.53 ± 0.03	3.12 ± 0.14	-2.57 ± 0.05	2.58 ± 0.03
MFVI	LRELU	1.51 ± 0.04	-2.09 ± 0.03	2.44 ± 0.03	2.74 ± 0.16	-2.39 ± 0.04	2.45 ± 0.03
	TANH	1.45 ± 0.04	-2.12 ± 0.03	2.50 ± 0.03	2.93 ± 0.13	-2.48 ± 0.04	2.50 ± 0.03
ENSEMBLE	LRELU	0.54 ± 0.02	-1.59 ± 0.04	2.05 ± 0.04	2.79 ± 0.17	-2.48 ± 0.09	2.18 ± 0.03
	TANH	1.09 ± 0.03	-1.79 ± 0.04	2.16 ± 0.04	2.71 ± 0.13	-2.48 ± 0.08	2.24 ± 0.03
DROPOUT	LRELU	1.33 ± 0.03	-2.03 ± 0.02	2.41 ± 0.02	2.78 ± 0.16	-2.36 ± 0.04	2.41 ± 0.02
	TANH	1.55 ± 0.03	-2.12 ± 0.01	2.48 ± 0.01	2.77 ± 0.15	-2.41 ± 0.04	2.48 ± 0.01
SWAG	LRELU	2.12 ± 0.10	-2.21 ± 0.05	2.41 ± 0.06	3.08 ± 0.35	-2.64 ± 0.16	2.41 ± 0.06
MAP	LRELU	0.64 ± 0.03	-2.09 ± 0.04	2.57 ± 0.04	3.02 ± 0.17	-2.60 ± 0.07	2.57 ± 0.04
	TANH	1.48 ± 0.03	-2.18 ± 0.03	2.58 ± 0.03	3.01 ± 0.17	-2.59 ± 0.06	2.58 ± 0.03

Table 8: Regression results for the concrete dataset, means and standard errors over 20 seeds

CONCRETE		TRAIN			TEST		
$n = 927,\ k = 8$		RMSE ↓	LLH ↑	ENTR ↓	RMSE ↓	LLH ↑	ENTR ↓
GP	RBF	3.41 ± 0.03	-2.79 ± 0.01	3.05 ± 0.01	5.62 ± 0.13	-3.08 ± 0.02	3.13 ± 0.01
GBLL	LRELU	1.60 ± 0.03	-2.74 ± 0.03	3.20 ± 0.03	5.01 ± 0.18	-3.09 ± 0.03	3.20 ± 0.03
	TANH	1.83 ± 0.03	-2.84 ± 0.03	3.30 ± 0.04	5.50 ± 0.23	-3.21 ± 0.03	3.30 ± 0.04
LDGBLL	LRELU	1.67 ± 0.03	-2.59 ± 0.02	3.03 ± 0.03	4.80 ± 0.18	-2.97 ± 0.03	3.05 ± 0.03
	TANH	1.70 ± 0.03	-2.50 ± 0.02	2.93 ± 0.02	4.39 ± 0.14	-2.89 ± 0.03	2.93 ± 0.02
MFVI	LRELU	3.04 ± 0.08	-2.62 ± 0.03	2.88 ± 0.03	4.80 ± 0.13	-2.97 ± 0.03	2.88 ± 0.03
	TANH	3.16 ± 0.09	-2.66 ± 0.03	2.92 ± 0.03	5.04 ± 0.12	-3.04 ± 0.02	2.92 ± 0.03
ENSEMBLE	LRELU	2.06 ± 0.08	-2.21 ± 0.05	2.47 ± 0.05	4.55 ± 0.12	-3.04 ± 0.08	2.55 ± 0.04
	TANH	2.37 ± 0.11	-2.29 ± 0.05	2.47 ± 0.04	4.51 ± 0.13	-3.03 ± 0.07	2.54 ± 0.04
DROPOUT	LRELU	2.60 ± 0.04	-2.65 ± 0.01	3.01 ± 0.01	4.45 ± 0.11	-2.90 ± 0.02	3.02 ± 0.01
	TANH	3.66 ± 0.01	-2.87 ± 0.00	3.17 ± 0.00	4.90 ± 0.10	-3.03 ± 0.01	3.17 ± 0.00
SWAG	LRELU	3.98 ± 0.09	-2.81 ± 0.02	2.90 ± 0.02	5.50 ± 0.16	-3.19 ± 0.05	2.90 ± 0.02
MAP	LRELU	2.48 ± 0.02	-2.49 ± 0.02	2.79 ± 0.02	4.75 ± 0.12	-3.04 ± 0.04	2.79 ± 0.02
	TANH	3.72 ± 0.02	-2.75 ± 0.01	2.88 ± 0.01	5.15 ± 0.13	-3.11 ± 0.04	2.88 ± 0.01

Table 9: Regression results for the energy dataset, means and standard errors over 20 seeds

ENERGY		TRAIN			TEST		
$n = 691,\ k = 8$		RMSE ↓	LLH ↑	ENTR ↓	RMSE ↓	LLH ↑	ENTR ↓
GP	RBF	0.30 ± 0.00	-0.30 ± 0.01	0.52 ± 0.01	0.47 ± 0.01	-0.66 ± 0.04	0.57 ± 0.01
GBLL	LRELU	0.16 ± 0.01	-0.34 ± 0.03	0.79 ± 0.03	0.46 ± 0.02	-0.69 ± 0.03	0.79 ± 0.03
	TANH	0.30 ± 0.01	-0.32 ± 0.03	0.58 ± 0.03	0.44 ± 0.02	-0.62 ± 0.04	0.58 ± 0.03
LDGBLL	LRELU	0.40 ± 0.01	-0.71 ± 0.03	1.03 ± 0.04	0.50 ± 0.02	-0.81 ± 0.03	1.04 ± 0.04
	TANH	0.41 ± 0.01	-0.67 ± 0.02	0.95 ± 0.02	0.53 ± 0.01	-0.81 ± 0.02	0.95 ± 0.02
MFVI	LRELU	0.29 ± 0.01	-0.31 ± 0.04	0.58 ± 0.05	0.43 ± 0.01	-0.63 ± 0.05	0.59 ± 0.05
	TANH	0.32 ± 0.01	-0.36 ± 0.03	0.59 ± 0.04	0.48 ± 0.01	-0.72 ± 0.04	0.60 ± 0.04
ENSEMBLE	LRELU	0.14 ± 0.00	0.28 ± 0.02	0.09 ± 0.02	0.41 ± 0.02	-0.58 ± 0.07	0.25 ± 0.01
	TANH	0.28 ± 0.00	-0.19 ± 0.01	0.39 ± 0.01	0.41 ± 0.01	-0.57 ± 0.06	0.41 ± 0.01
DROPOUT	LRELU	0.41 ± 0.00	-1.30 ± 0.00	1.76 ± 0.00	0.53 ± 0.01	-1.33 ± 0.00	1.76 ± 0.00
	TANH	0.59 ± 0.00	-1.46 ± 0.00	1.90 ± 0.00	0.66 ± 0.01	-1.48 ± 0.00	1.90 ± 0.00
SWAG	LRELU	0.78 ± 0.10	-1.16 ± 0.10	1.44 ± 0.10	0.93 ± 0.09	-1.29 ± 0.08	1.40 ± 0.10
MAP	LRELU	0.15 ± 0.00	0.20 ± 0.02	0.16 ± 0.02	0.53 ± 0.01	-1.44 ± 0.09	0.16 ± 0.02
	TANH	0.32 ± 0.00	-0.27 ± 0.01	0.31 ± 0.01	0.45 ± 0.01	-0.79 ± 0.06	0.31 ± 0.01

Table 10: Regression results for the kin8nm dataset, means and standard errors over 20 seeds

KIN8NM		TRAIN			TEST		
$n = 7373,\ k = 8$		RMSE ↓	LLH ↑	ENTR ↓	RMSE ↓	LLH ↑	ENTR ↓
GP	RBF	0.04 ± 0.00	1.73 ± 0.01	-1.60 ± 0.03	0.07 ± 0.00	1.10 ± 0.04	-1.49 ± 0.03
GBLL	LRELU	0.04 ± 0.00	1.40 ± 0.01	-1.01 ± 0.01	0.08 ± 0.00	1.11 ± 0.01	-1.01 ± 0.01
	TANH	0.05 ± 0.00	1.45 ± 0.01	-1.16 ± 0.01	0.07 ± 0.00	1.27 ± 0.01	-1.16 ± 0.01
LDGBLL	LRELU	0.05 ± 0.00	1.41 ± 0.00	-1.17 ± 0.01	0.07 ± 0.00	1.23 ± 0.01	-1.17 ± 0.01
	TANH	0.05 ± 0.00	1.43 ± 0.00	-1.19 ± 0.00	0.07 ± 0.00	1.29 ± 0.00	-1.19 ± 0.00
MFVI	LRELU	0.06 ± 0.00	1.38 ± 0.01	-1.26 ± 0.01	0.07 ± 0.00	1.23 ± 0.01	-1.25 ± 0.01
	TANH	0.06 ± 0.00	1.35 ± 0.01	-1.22 ± 0.01	0.07 ± 0.00	1.21 ± 0.01	-1.22 ± 0.01
ENSEMBLE	LRELU	0.05 ± 0.00	1.59 ± 0.01	-1.38 ± 0.01	0.06 ± 0.00	1.33 ± 0.01	-1.35 ± 0.01
	TANH	0.05 ± 0.00	1.52 ± 0.00	-1.38 ± 0.00	0.06 ± 0.00	1.34 ± 0.01	-1.36 ± 0.00
DROPOUT	LRELU	0.07 ± 0.00	1.11 ± 0.00	-0.80 ± 0.00	0.08 ± 0.00	1.06 ± 0.00	-0.80 ± 0.00
	TANH	0.08 ± 0.00	0.93 ± 0.00	-0.65 ± 0.00	0.09 ± 0.00	0.91 ± 0.00	-0.65 ± 0.00
SWAG	LRELU	0.06 ± 0.00	1.40 ± 0.01	-1.37 ± 0.01	0.07 ± 0.00	1.16 ± 0.01	-1.37 ± 0.01
MAP	LRELU	0.06 ± 0.00	1.41 ± 0.01	-1.40 ± 0.01	0.07 ± 0.00	1.18 ± 0.01	-1.40 ± 0.01
	TANH	0.06 ± 0.00	1.42 ± 0.01	-1.41 ± 0.01	0.07 ± 0.00	1.27 ± 0.01	-1.41 ± 0.01

Table 11: Regression results for the naval dataset, means and standard errors over 20 seeds

NAVAL		TRAIN			TEST		
$n = 10741,\ k = 16$		RMSE ↓	LLH ↑	ENTR ↓	RMSE ↓	LLH ↑	ENTR ↓
GP	RBF	0.00 ± 0.00	4.64 ± 0.01	-4.14 ± 0.01	0.00 ± 0.00	4.64 ± 0.01	-4.14 ± 0.01
GBLL	LRELU	0.00 ± 0.00	6.79 ± 0.29	-6.78 ± 0.00	0.00 ± 0.00	6.77 ± 0.29	-6.78 ± 0.00
	TANH	0.00 ± 0.00	6.51 ± 0.03	-6.39 ± 0.02	0.00 ± 0.00	6.49 ± 0.03	-6.39 ± 0.02
LDGBLL	LRELU	0.00 ± 0.00	5.41 ± 0.03	-4.94 ± 0.03	0.00 ± 0.00	5.41 ± 0.03	-4.94 ± 0.03
	TANH	0.00 ± 0.00	5.76 ± 0.03	-5.35 ± 0.03	0.00 ± 0.00	5.75 ± 0.03	-5.35 ± 0.03
MFVI	LRELU	0.00 ± 0.00	8.34 ± 0.03	-8.33 ± 0.03	0.00 ± 0.00	7.96 ± 0.02	-8.33 ± 0.03
	TANH	0.00 ± 0.00	8.40 ± 0.04	-8.31 ± 0.04	0.00 ± 0.00	7.81 ± 0.35	-8.31 ± 0.04
ENSEMBLE	LRELU	0.00 ± 0.00	7.77 ± 0.01	-7.36 ± 0.01	0.00 ± 0.00	7.74 ± 0.01	-7.36 ± 0.01
	TANH	0.00 ± 0.00	7.72 ± 0.01	-7.31 ± 0.01	0.00 ± 0.00	7.70 ± 0.01	-7.30 ± 0.01
DROPOUT	LRELU	0.00 ± 0.00	5.20 ± 0.00	-4.72 ± 0.00	0.00 ± 0.00	5.19 ± 0.00	-4.72 ± 0.00
	TANH	0.00 ± 0.00	5.01 ± 0.00	-4.55 ± 0.00	0.00 ± 0.00	5.01 ± 0.00	-4.55 ± 0.00
SWAG	LRELU	0.00 ± 0.00	5.60 ± 0.07	-5.20 ± 0.07	0.00 ± 0.00	5.61 ± 0.06	-5.20 ± 0.07
MAP	LRELU	0.00 ± 0.00	8.48 ± 0.02	-8.45 ± 0.02	0.00 ± 0.00	8.01 ± 0.05	-8.45 ± 0.02
	TANH	0.00 ± 0.00	8.83 ± 0.03	-8.84 ± 0.01	0.00 ± 0.00	8.74 ± 0.04	-8.84 ± 0.01

Table 12: Regression results for the power dataset, means and standard errors over 20 seeds

POWER		TRAIN			TEST		
$n = 8611,\ k = 4$		RMSE ↓	LLH ↑	ENTR ↓	RMSE ↓	LLH ↑	ENTR ↓
GP	RBF	3.30 ± 0.01	-2.68 ± 0.00	2.87 ± 0.00	3.72 ± 0.04	-2.76 ± 0.01	2.88 ± 0.00
GBLL	LRELU	3.17 ± 0.02	-2.62 ± 0.01	2.80 ± 0.01	3.85 ± 0.03	-2.77 ± 0.01	2.80 ± 0.01
	TANH	4.01 ± 0.01	-2.81 ± 0.00	2.83 ± 0.01	4.09 ± 0.04	-2.83 ± 0.01	2.83 ± 0.01
LDGBLL	LRELU	3.45 ± 0.04	-2.68 ± 0.01	2.80 ± 0.01	3.85 ± 0.04	-2.77 ± 0.01	2.80 ± 0.01
	TANH	3.94 ± 0.01	-2.80 ± 0.00	2.88 ± 0.00	4.05 ± 0.04	-2.82 ± 0.01	2.88 ± 0.00
MFVI	LRELU	3.65 ± 0.02	-2.72 ± 0.00	2.74 ± 0.00	3.86 ± 0.04	-2.77 ± 0.01	2.74 ± 0.00
	TANH	3.77 ± 0.01	-2.75 ± 0.00	2.78 ± 0.01	3.91 ± 0.04	-2.79 ± 0.01	2.78 ± 0.01
ENSEMBLE	LRELU	3.07 ± 0.01	-2.55 ± 0.00	2.66 ± 0.00	3.59 ± 0.04	-2.70 ± 0.01	2.67 ± 0.00
	TANH	3.29 ± 0.01	-2.61 ± 0.00	2.69 ± 0.00	3.66 ± 0.04	-2.72 ± 0.01	2.69 ± 0.00
DROPOUT	LRELU	3.77 ± 0.01	-2.78 ± 0.00	2.94 ± 0.00	3.90 ± 0.04	-2.80 ± 0.01	2.94 ± 0.00
	TANH	4.12 ± 0.01	-2.85 ± 0.00	2.98 ± 0.00	4.18 ± 0.03	-2.86 ± 0.01	2.98 ± 0.00
SWAG	LRELU	3.41 ± 0.03	-2.65 ± 0.01	2.70 ± 0.00	3.85 ± 0.05	-2.77 ± 0.02	2.70 ± 0.00
MAP	LRELU	3.47 ± 0.02	-2.66 ± 0.01	2.67 ± 0.01	3.81 ± 0.04	-2.77 ± 0.01	2.67 ± 0.01
	TANH	3.44 ± 0.03	-2.65 ± 0.01	2.66 ± 0.01	3.78 ± 0.04	-2.76 ± 0.01	2.66 ± 0.01

Table 17: Number of hidden units, learning rates and maximum number of epochs for standard regression tasks.

STANDARD		BOSTON	CONCRETE	ENERGY	KIN8NM	NAVAL	POWER	PROTEIN	WINE	YACHT	SARCOS
HIDDEN DIMS		50 50	50 50	50 50	50 50	50 50	50 50	50 50	50 50	50 50	50 200 200
GBLL	LRELU	1E-3, 3000	1E-3, 3000	1E-3, 8000	1E-3, 3000	1E-3, 8000	1E-3, 5000	1E-3, 2000	1E-4, 1000	1E-3, 8000	2E-4 30000
	TANH	1E-3, 2000	1E-3, 3000	1E-3, 8000	1E-3, 3000	1E-3, 8000	1E-3, 5000	1E-3, 2000	1E-4, 1000	1E-3, 8000	2E-4, 30000
LDGBLL	LRELU	1E-3, 4000	1E-3, 4000	1E-3, 10000	1E-3, 5000	1E-3, 10000	1E-3, 5000	1E-3, 5000	1E-4, 1000	1E-3, 10000	2E-4, 30000
	TANH	1E-3, 4000	1E-3, 4000	1E-3, 10000	1E-3, 5000	1E-3, 10000	1E-3, 5000	1E-3, 5000	1E-4, 1000	1E-3, 10000	2E-4, 30000
MFVI	LRELU	1E-3, 10000	1E-3, 15000	1E-3, 30000	1E-3, 20000	1E-3, 100000	1E-3, 20000	1E-3, 30000	1E-4, 20000	1E-3, 20000	5E-3, 10000
	TANH	1E-3, 10000	1E-3, 15000	1E-3, 30000	1E-3, 20000	1E-3, 100000	1E-3, 20000	1E-3, 30000	1E-4, 20000	1E-3, 20000	5E-3, 10000
DROPOUT	LRELU	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 5000
	TANH	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 5000
ENSEMBLE	LRELU	1E-3, 500	1E-3, 500	1E-3, 600	1E-3, 500	1E-3, 500	1E-3, 500	1E-3, 300	1E-4, 500	1E-3, 1200	1E-4, 3000
	TANH	1E-3, 500	1E-3, 500	1E-3, 600	1E-3, 300	1E-3, 500	1E-3, 500	1E-3, 300	1E-4, 500	1E-3, 1200	1E-4, 3000
SWAG	LRELU	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-5, 4000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 5000
MAP	LRELU	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 5000
	TANH	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 10000	1E-4, 5000
GP	RBF	1E-2, 1000	1E-2, 1000	1E-2, 2000	1E-2, 1000	1E-3, 3000	1E-2, 1000	1E-3, 3000	1E-2, 1000	1E-2, 2000	1E-2, 5000

Table 13: Regression results for the protein dataset, means and standard errors over 20 seeds

PROTEIN		TRAIN			TEST		
$n = 41157,\ k = 9$		RMSE ↓	LLH ↑	ENTR ↓	RMSE ↓	LLH ↑	ENTR ↓
GP	RBF	2.84 ± 0.02	-2.86 ± 0.00	3.25 ± 0.00	3.60 ± 0.01	-2.92 ± 0.00	3.25 ± 0.00
GBLL	LRELU	3.40 ± 0.01	-2.67 ± 0.00	2.82 ± 0.01	4.00 ± 0.02	-2.81 ± 0.00	2.82 ± 0.01
	TANH	3.48 ± 0.03	-2.68 ± 0.01	2.78 ± 0.01	3.95 ± 0.02	-2.79 ± 0.01	2.78 ± 0.01
LDGBLL	LRELU	3.53 ± 0.02	-2.70 ± 0.00	2.84 ± 0.00	3.94 ± 0.02	-2.79 ± 0.00	2.84 ± 0.00
	TANH	3.43 ± 0.01	-2.68 ± 0.00	2.82 ± 0.00	3.86 ± 0.02	-2.77 ± 0.00	2.82 ± 0.00
MFVI	LRELU	3.49 ± 0.03	-2.67 ± 0.01	2.70 ± 0.01	3.86 ± 0.02	-2.77 ± 0.00	2.70 ± 0.01
	TANH	3.50 ± 0.03	-2.67 ± 0.01	2.68 ± 0.01	3.90 ± 0.02	-2.79 ± 0.00	2.68 ± 0.01
ENSEMBLE	LRELU	3.11 ± 0.01	-2.57 ± 0.00	2.71 ± 0.00	3.58 ± 0.01	-2.68 ± 0.00	2.73 ± 0.00
	TANH	3.09 ± 0.00	-2.57 ± 0.00	2.73 ± 0.00	3.58 ± 0.01	-2.67 ± 0.00	2.75 ± 0.00
DROPOUT	LRELU	3.97 ± 0.00	-2.81 ± 0.00	2.91 ± 0.00	4.09 ± 0.01	-2.83 ± 0.00	2.91 ± 0.00
	TANH	4.46 ± 0.00	-2.92 ± 0.00	2.98 ± 0.00	4.52 ± 0.01	-2.93 ± 0.00	2.98 ± 0.00
SWAG	LRELU	3.60 ± 0.03	-2.70 ± 0.01	2.72 ± 0.01	3.98 ± 0.01	-2.80 ± 0.00	2.72 ± 0.01
MAP	LRELU	3.54 ± 0.04	-2.68 ± 0.01	2.69 ± 0.01	3.93 ± 0.02	-2.80 ± 0.01	2.69 ± 0.01
	TANH	3.43 ± 0.03	-2.65 ± 0.01	2.66 ± 0.01	3.84 ± 0.02	-2.78 ± 0.01	2.66 ± 0.01

Table 14: Regression results for the wine dataset, means and standard errors over 20 seeds

WINE		TRAIN			TEST		
$n = 1439,\ k = 11$		RMSE ↓	LLH ↑	ENTR ↓	RMSE ↓	LLH ↑	ENTR ↓
GP	RBF	0.04 ± 0.02	1.05 ± 0.14	-0.57 ± 0.13	0.56 ± 0.01	-0.45 ± 0.05	0.55 ± 0.04
GBLL	LRELU	0.52 ± 0.02	-0.91 ± 0.02	1.20 ± 0.00	0.64 ± 0.01	-1.02 ± 0.01	1.20 ± 0.00
	TANH	0.55 ± 0.01	-0.94 ± 0.01	1.20 ± 0.00	0.64 ± 0.01	-1.01 ± 0.01	1.20 ± 0.00
LDGBLL	LRELU	0.59 ± 0.01	-0.98 ± 0.01	1.22 ± 0.00	0.64 ± 0.01	-1.02 ± 0.01	1.22 ± 0.00
	TANH	0.60 ± 0.01	-0.99 ± 0.00	1.23 ± 0.00	0.63 ± 0.01	-1.02 ± 0.01	1.23 ± 0.00
MFVI	LRELU	0.55 ± 0.00	-0.84 ± 0.01	0.93 ± 0.01	0.63 ± 0.01	-0.95 ± 0.01	0.94 ± 0.01
	TANH	0.59 ± 0.00	-0.91 ± 0.00	0.97 ± 0.01	0.63 ± 0.01	-0.97 ± 0.01	0.97 ± 0.01
ENSEMBLE	LRELU	0.55 ± 0.01	-0.82 ± 0.01	0.89 ± 0.02	0.62 ± 0.01	-0.95 ± 0.01	0.89 ± 0.01
	TANH	0.57 ± 0.00	-0.86 ± 0.01	0.89 ± 0.01	0.62 ± 0.01	-0.95 ± 0.01	0.90 ± 0.01
DROPOUT	LRELU	0.48 ± 0.01	-0.70 ± 0.03	0.85 ± 0.01	0.61 ± 0.01	-0.93 ± 0.01	0.85 ± 0.01
	TANH	0.54 ± 0.01	-0.81 ± 0.02	0.91 ± 0.01	0.62 ± 0.01	-0.94 ± 0.01	0.91 ± 0.01
SWAG	LRELU	0.57 ± 0.00	-0.86 ± 0.01	0.87 ± 0.01	0.63 ± 0.01	-0.96 ± 0.03	0.87 ± 0.01
MAP	LRELU	0.55 ± 0.00	-0.83 ± 0.01	0.91 ± 0.01	0.63 ± 0.01	-0.96 ± 0.01	0.91 ± 0.01
	TANH	0.58 ± 0.00	-0.87 ± 0.01	0.91 ± 0.01	0.63 ± 0.01	-0.96 ± 0.01	0.91 ± 0.01

Flight Delay

The flight delay dataset has 700k training points and 100k test points, predicting the delay time in minutes using features describing the flight and aircraft. Due to the large-scale nature of the task, we report the results of previous work. Note that the noise contrastive prior MFVI+NCP baseline used network of size $[1000, 1000, 1000]$. The deep Gaussian process is for the (superior) 5 layer model.

We observed competitive performance with a MAP baseline with an architecture of only $[100, 100, 100]$, and this smaller size allowed us to compare the exact and variational BLL within the GPU memory limits (32GB), as there was a memory bottleneck in computing the model Jacobians.

Table 15: Regression results for the yacht dataset, means and standard errors over 20 seeds

YACHT		TRAIN			TEST		
$n = 277,\ k = 6$		RMSE ↓	LLH ↑	ENTR ↓	RMSE ↓	LLH ↑	ENTR ↓
GP	RBF	0.13 ± 0.00	0.34 ± 0.01	0.02 ± 0.01	0.40 ± 0.03	-0.17 ± 0.03	0.23 ± 0.02
GBLL	LRELU	0.06 ± 0.00	-1.05 ± 0.05	1.55 ± 0.05	1.09 ± 0.09	-1.67 ± 0.11	1.55 ± 0.05
	TANH	0.24 ± 0.02	-0.20 ± 0.09	0.43 ± 0.03	0.43 ± 0.03	-0.70 ± 0.10	0.43 ± 0.03
LDGBLL	LRELU	0.28 ± 0.01	-0.89 ± 0.04	1.34 ± 0.04	0.75 ± 0.10	-1.13 ± 0.06	1.37 ± 0.04
	TANH	0.23 ± 0.01	-0.48 ± 0.03	0.91 ± 0.03	0.52 ± 0.05	-0.73 ± 0.05	0.93 ± 0.03
MFVI	LRELU	0.26 ± 0.02	-0.64 ± 0.07	1.08 ± 0.07	1.10 ± 0.11	-1.43 ± 0.17	1.08 ± 0.07
	TANH	0.31 ± 0.02	-0.76 ± 0.08	1.19 ± 0.09	1.26 ± 0.14	-1.44 ± 0.15	1.21 ± 0.09
ENSEMBLE	LRELU	0.08 ± 0.01	0.36 ± 0.03	0.12 ± 0.03	0.83 ± 0.08	-0.35 ± 0.07	0.38 ± 0.02
	TANH	0.12 ± 0.00	0.32 ± 0.01	0.10 ± 0.02	0.38 ± 0.03	-0.03 ± 0.05	0.14 ± 0.01
DROPOUT	LRELU	0.44 ± 0.01	-1.77 ± 0.00	2.26 ± 0.00	1.21 ± 0.13	-1.82 ± 0.01	2.26 ± 0.01
	TANH	1.20 ± 0.01	-2.22 ± 0.00	2.66 ± 0.00	1.20 ± 0.11	-2.24 ± 0.01	2.67 ± 0.00
SWAG	LRELU	0.85 ± 0.13	-1.02 ± 0.07	1.41 ± 0.05	1.13 ± 0.20	-1.11 ± 0.05	1.33 ± 0.04
MAP	LRELU	0.03 ± 0.00	-0.15 ± 0.14	0.65 ± 0.14	0.94 ± 0.09	-5.14 ± 1.62	0.65 ± 0.14
	TANH	0.08 ± 0.00	0.45 ± 0.10	-0.02 ± 0.11	0.39 ± 0.04	-1.77 ± 0.53	-0.02 ± 0.11

Table 16: Regression results for the sarcoss dataset, means and standard errors over 20 seeds

SARCOS		TRAIN			TEST		
$n = 35000,\ k = 21$		RMSE ↓	LLH ↑	ENTR ↓	RMSE ↓	LLH ↑	ENTR ↓
GP	RBF	1.66 ± 0.00	-2.16 ± 0.00	2.49 ± 0.00	1.69 ± 0.01	-2.18 ± 0.00	2.50 ± 0.00
GBLL	LRELU	0.76 ± 0.02	-2.03 ± 0.00	2.50 ± 0.01	1.37 ± 0.02	-2.11 ± 0.00	2.50 ± 0.01
	TANH	2.19 ± 0.03	-2.21 ± 0.01	2.23 ± 0.02	2.19 ± 0.03	-2.22 ± 0.01	2.23 ± 0.02
LDGBLL	LRELU	3.66 ± 0.06	-3.45 ± 0.01	3.90 ± 0.01	3.62 ± 0.06	-3.45 ± 0.01	3.90 ± 0.01
	TANH	3.30 ± 0.07	-3.08 ± 0.02	3.50 ± 0.02	3.26 ± 0.08	-3.08 ± 0.02	3.50 ± 0.02
MFVI	LRELU	4.17 ± 0.03	-2.86 ± 0.01	2.61 ± 0.03	4.13 ± 0.03	-2.83 ± 0.01	2.61 ± 0.03
	TANH	3.84 ± 0.14	-2.89 ± 0.05	2.45 ± 0.05	3.84 ± 0.14	-2.90 ± 0.05	2.45 ± 0.05
ENSEMBLE	LRELU	1.09 ± 0.02	-1.57 ± 0.01	1.81 ± 0.01	1.28 ± 0.01	-1.67 ± 0.01	1.83 ± 0.01
	TANH	1.25 ± 0.01	-1.67 ± 0.01	1.87 ± 0.01	1.38 ± 0.01	-1.76 ± 0.01	1.88 ± 0.01
DROPOUT	LRELU	1.77 ± 0.00	-2.19 ± 0.00	2.53 ± 0.00	1.81 ± 0.01	-2.19 ± 0.00	2.53 ± 0.00
	TANH	2.85 ± 0.01	-2.57 ± 0.00	2.85 ± 0.00	2.83 ± 0.01	-2.56 ± 0.00	2.85 ± 0.00
SWAG	LRELU	3.16 ± 0.05	-2.59 ± 0.02	2.27 ± 0.03	3.14 ± 0.06	-2.58 ± 0.02	2.28 ± 0.03
MAP	LRELU	1.40 ± 0.00	-1.76 ± 0.00	1.80 ± 0.00	1.54 ± 0.01	-1.88 ± 0.01	1.80 ± 0.00
	TANH	1.82 ± 0.00	-2.02 ± 0.00	2.03 ± 0.00	1.84 ± 0.00	-2.05 ± 0.00	2.03 ± 0.00

I.2 Active learning

The cartpole dataset in full is displayed in Figure 11. The regression task was for inverse dynamics, mapping the dynamic state ($k = 6$) to the recorded drive torque. To construct the test data, every 5th sample was extracted from the first half of the data, therefore containing approximately 50 / 50 swing-up and stabilization samples.

For active learning, following previous work [27] an information-based acquisition rule was used, following based on a Bayesian model's information gain [42]

$$\text{Infogain}(\mathbf{x}) = 0.5 \log\left(1 + \frac{\sigma_e^2(\mathbf{x})}{\sigma_a^2(\mathbf{x})}\right). \tag{65}$$

When selecting several points at a time, as this rule can have several local maxima it is beneficial to use this rule

Table 18: Regression results for the flight dataset, means and standard errors over 10 seeds

FLIGHT DELAY		TRAIN			TEST		
$n = 70000,\ k = 8$		RMSE ↓	LLH ↑	ENTR ↓	RMSE ↓	LLH ↑	ENTR ↓
MAP	TANH	23.56 ± 0.03	-4.58 ± 0.00	4.56 ± 0.00	24.20 ± 0.02	-4.61 ± 0.00	4.56 ± 0.00
GBLL	TANH	26.75 ± 0.09	-4.71 ± 0.00	4.75 ± 0.00	27.00 ± 0.08	-4.72 ± 0.00	4.75 ± 0.00
LDGBLL	TANH	26.60 ± 0.17	-4.70 ± 0.01	4.75 ± 0.00	26.87 ± 0.17	-4.72 ± 0.01	4.75 ± 0.00
VAR. GBLL	TANH	30.32 ± 0.00	-4.74 ± 0.00	4.71 ± 0.01	30.47 ± 0.00	-4.76 ± 0.00	4.71 ± 0.01
VAR. LDGBLL	TANH	30.33 ± 0.00	-4.74 ± 0.00	4.82 ± 0.01	30.48 ± 0.00	-4.75 ± 0.00	4.82 ± 0.01
SVI GP[29]		-	-	-	32.6	-	-
DGP 5[67]		-	-	-	24.1	-4.58	-
MFVI+NCP[27]	LRELU	-	-	-	24.71	-4.38	-

as a categorical distribution

$$\text{Cat}(\mathbf{X}) = \text{Softmax}(\alpha \, \text{Infogain}(\mathbf{X})). \tag{66}$$

By applying a softmax transformation to 65, datapoints can be sampled without replacement. The temperature α was needed to shape the modes of the distribution. To be robust across dataset sizes and Bayesian models, it was set to be $\alpha = 0.01n/\max(\text{Infogain}(\mathbf{X}))$, where n is the number of points.

For the cartpole experiment, starting with 25 randomly drawn samples, 25 additional points were selected over 40 iterations. The complete cartpole dataset contains 7490 points, collected at 250Hz. To construct a fair test set (over swing-up and stabilization), it was constructed by downsampling the first half of the data by a factor of 5. Therefore the available training data consisted of 6741 points, and the test 749.

Table 19: Active learning task hyperparameters

	ITERS.	NUM. START	NUM. ACQUIRE
CARTPOLE	40	25	25

Table 20: Active learning model hyperparameters

		LEARNING RATE	ITERATIONS
TBLL	TANH, $[50, 250]$, 1	1E-3	1000
LDTBLL	TANH, $[50, 250]$, 1	1E-4	1000
MFVI	TANH, $[50, 250]$, 1	1E-3	5000

I.3 Bayesian optimization

The experiments are performed using `BoTorch` library [2]. We used their default `SingleTaskGp`, which uses a Matérn 5/2 kernel with an inverse gamma prior. The EI acquisition metric was optimizing 'jointly' using the default setting of BoTorch's `optimize_acqf` function. Task hyperparameters are detailed in Table 21, and the models are described in Table 22.

Table 21: Bayesian optimization task hyperparameters

	ITERS.	STARTING SAMPLES	ACQ. SAMPLES	ACQ. RESTARTS
SINC IN A HAYSTACK	40	4	500	100
HARTMAN6	100	10	5000	1000

Table 22: Bayesian optimization model hyperparameters

	SINC IN A HAYSTACK			HARTMANN6		
		LR.	ITERS.		LR.	ITERS.
GP	MATÉRN	1E-3	1000	MATÉRN	1E-3	1000
TBLL	TANH, $[50, 50]$, 1	1E-3	1000	TANH, $[50, 100]$, 10	1E-3	1000
LDTBLL	TANH, $[50, 50]$, 1	1E-4	1000	TANH, $[50, 100]$, 10	1E-4	1000