Deep Sigma Point Processes

Martin Jankowiak[*]
The Broad Institute

Geoff Pleiss
Cornell University

Jacob R. Gardner
University of Pennsylvania

Abstract

We introduce Deep Sigma Point Processes, a class of parametric models inspired by the compositional structure of Deep Gaussian Processes (DGPs). Deep Sigma Point Processes (DSPPs) retain many of the attractive features of (variational) DGPs, including mini-batch training and predictive uncertainty that is controlled by kernel basis functions. Importantly, since DSPPs admit a simple maximum likelihood inference procedure, the resulting predictive distributions are not degraded by any posterior approximations. In an extensive empirical comparison on univariate and multivariate regression tasks we find that the resulting predictive distributions are significantly better calibrated than those obtained with other probabilistic methods for scalable regression, including variational DGPs—often by as much as a nat per datapoint.

1 INTRODUCTION

As machine learning becomes utilized for increasingly high risk applications such as medical treatment suggestion and autonomous driving, calibrated uncertainty estimation becomes critical. As a result, substantial effort has been devoted to the development of flexible probabilistic models. Gaussian Processes (GPs) have emerged as an important class of models in cases where predictive uncertainty estimates are essential (Rasmussen, 2003). Unfortunately, the simplest GP models often fail to match the expressive power of their neural network counterparts.

This limited flexibility has motivated the introduction of Deep Gaussian Processes (DGPs), which compose multiple layers of latent functions to build up more flexible function priors (Damianou and Lawrence, 2013). While this class of models has shown considerable promise, inference remains a serious challenge, with empirical evidence suggesting—not surprisingly—that posterior approximations (e.g. factorizing across layers) can degrade the performance of the predictive distribution (Havasi et al., 2018). Indeed, targeting posterior approximations as an optimization objective may fail both to achieve good predictive performance and to faithfully approximate the exact posterior. Clearly, this motivates investigating alternative approaches.

In this work we take a different approach to constructing flexible probabilistic models. In particular we formulate a class of *fully parametric* models that retain many of the attractive features of variational deep GPs, while posing a much easier (maximum likelihood) inference problem. This pragmatic approach is motivated by the recognition that—especially in settings where predictive performance is paramount—it is essential to consider the interplay between modeling and inference. In particular, as motivated above, a compelling class of models may be of limited predictive utility if approximations in the inference procedure severely degrade the posterior predictive distribution. Conversely, it can be a significant advantage if a class of models admits a simple inference procedure.

Similarly to variational deep GPs, our regression models utilize predictive distributions that are mixtures of Normal distributions whose mean and variance functions make use of hierarchical composition and kernel interpolation. In contrast to deep GPs, however, our parametric perspective allows us to directly target the predictive distribution in the training objective. As we show empirically in Sec. 5 the model we introduce—the *Deep Sigma Point Process* (DSPP)—exhibits excellent predictive performance and dramatically outperforms a number of strong baselines, including Deep Kernel Learning (Calandra et al., 2016; Wilson et al., 2016) and variational Deep Gaussian Processes (Salimbeni and Deisenroth, 2017).

[*] Correspondence to: jankowiak@gmail.com. This work was completed while MJ and JG were at Uber AI.

Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), PMLR volume 124, 2020.

2 BACKGROUND

This section is organized as follows. In Sec. 2.1-2.2 we review the basics of Gaussian Processes and inducing point methods. In Sec. 2.3 we review Deep Gaussian Processes. In Sec. 2.4 we review PPGPR (Jankowiak et al., 2020), as it serves as motivation for DSPPs in Sec. 3. We also use this section to establish our notation.

2.1 GAUSSIAN PROCESS REGRESSION

In probabilistic modeling Gaussian Processes offer flexible non-parametric function priors that are useful in various regression and classification tasks (Rasmussen, 2003). For a given input space $\mathbb{R}^d$ GPs are entirely specified by a kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ and a mean function $\mu : \mathbb{R}^d \to \mathbb{R}$. Different choices of μ and k permit the modeler to encode prior information about the generative process. In the prototypical case of univariate regression[1] the joint density takes the form

$$p(\mathbf{y}, \mathbf{f}|\mathbf{X}) = p(\mathbf{y}|\mathbf{f}, \sigma_{\text{obs}}^2) p(\mathbf{f}|\mathbf{X}) \tag{1}$$

where $\mathbf{y}$ are the real-valued targets, $\mathbf{f}$ are the latent function values, $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ are the N inputs with $\mathbf{x}_i \in \mathbb{R}^d$, $p(\mathbf{f}|\mathbf{X})$ is a multivariate Normal distribution with covariance $\mathbf{K}_{NN} = k(\mathbf{X}, \mathbf{X})$, and σ_{obs}^2 is the variance of the Normal likelihood $p(\mathbf{y}|\cdot)$. The marginal likelihood takes the form

$$p(\mathbf{y}|\mathbf{X}) = \int d\mathbf{f} \; p(\mathbf{y}|\mathbf{f}, \sigma_{\text{obs}}^2) p(\mathbf{f}|\mathbf{X}) \tag{2}$$

Eqn. 2 can be computed analytically, but doing so is computationally prohibitive for large datasets, necessitating approximate methods when N is large.

2.2 SPARSE GAUSSIAN PROCESSES

Recent years have seen significant progress in scaling Gaussian Process inference to large datasets. This advance has been enabled by the development of inducing point methods (Snelson and Ghahramani, 2006; Titsias, 2009; Hensman et al., 2013), which we now review. One begins by introducing inducing variables $\mathbf{u}$ that depend on variational parameters $\{\mathbf{z}_m\}_{m=1}^M$, with each $\mathbf{z}_m \in \mathbb{R}^d$ and where $M = \dim(\mathbf{u}) \ll N$. One then augments the GP prior with the auxiliary variables $\mathbf{u}$

$$p(\mathbf{f}|\mathbf{X}) \to p(\mathbf{f}|\mathbf{u}, \mathbf{X}, \mathbf{Z}) p(\mathbf{u}|\mathbf{Z})$$

and then appeals to Jensen's inequality to lower bound the log joint density over the inducing variables and the targets:

$$\begin{aligned} \log p(\mathbf{y}, \mathbf{u}|\mathbf{X}, \mathbf{Z}) &= \log \int d\mathbf{f} \, p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\mathbf{u}) p(\mathbf{u}) \\ &\geq \mathbb{E}_{p(\mathbf{f}|\mathbf{u})} \left[\log p(\mathbf{y}|\mathbf{f}) + \log p(\mathbf{u})\right] \\ &= \sum_{i=1}^N \log \mathcal{N}(y_i | \mathbf{k}_i^T \mathbf{K}_{MM}^{-1} \mathbf{u}, \sigma_{\text{obs}}^2) \\ &\quad - \tfrac{1}{2\sigma_{\text{obs}}^2} \operatorname{Tr} \widetilde{\mathbf{K}}_{NN} + \log p(\mathbf{u}) \end{aligned} \tag{3}$$

where $\widetilde{\mathbf{K}}_{NN}$ is given by

$$\widetilde{\mathbf{K}}_{NN} = \mathbf{K}_{NN} - \mathbf{K}_{NM} \mathbf{K}_{MM}^{-1} \mathbf{K}_{MN} \tag{4}$$

and

$$\begin{aligned} \mathbf{K}_{MM} &= k(\mathbf{Z}, \mathbf{Z}) \qquad \mathbf{k}_i = k(\mathbf{x}_i, \mathbf{Z}) \\ \mathbf{K}_{NM} &= \mathbf{K}_{MN}^{\mathrm{T}} = k(\mathbf{X}, \mathbf{Z}) \end{aligned} \tag{5}$$

Eqn. 3 can be used to construct a variety of algorithms for scalable GP inference; here we limit our discussion to SVGP (Hensman et al., 2013).

2.2.1 VARIATIONAL INFERENCE: SVGP

If we apply standard techniques from variational inference to the lower bound in Eqn. 3, we obtain SVGP, a popular algorithm for scalable GP inference. In more detail, SVGP proceeds by introducing a multivariate Normal variational distribution $q(\mathbf{u}) = \mathcal{N}(\boldsymbol{m}, \mathbf{S})$ and computing the ELBO, which is the expectation of Eqn. 3 w.r.t. $q(\mathbf{u})$ together with an entropy term term $H[q(\mathbf{u})]$:

$$\begin{aligned} &\mathcal{L}_{\text{svgp}} = \mathbb{E}_{q(\mathbf{u})} \left[\log p(\mathbf{y}, \mathbf{u}|\mathbf{X}, \mathbf{Z})\right] + H[q(\mathbf{u})] \\ &= \sum_{i=1}^N \left\{ \log \mathcal{N}(y_i | \mu_f(\mathbf{x}_i), \sigma_{\text{obs}}^2) - \frac{1}{2} \frac{\sigma_f(\mathbf{x}_i)^2}{\sigma_{\text{obs}}^2} \right\} \\ &- \mathrm{KL}(q(\mathbf{u}) | p(\mathbf{u})) \end{aligned} \tag{6}$$

Here KL denotes the Kullback-Leibler divergence, $\mu_f(\mathbf{x}_i)$ is the predictive mean function given by

$$\mu_f(\mathbf{x}_i) = \mathbf{k}_i^T \mathbf{K}_{MM}^{-1} \boldsymbol{m} \tag{7}$$

and $\sigma_f(\mathbf{x}_i)^2 \equiv \mathrm{Var}[\mathrm{f_i}|\mathrm{x_i}]$ denotes latent function variance

$$\sigma_f(\mathbf{x}_i)^2 = \widetilde{\mathbf{K}}_{ii} + \mathbf{k}_i^T \mathbf{K}_{MM}^{-1} \mathbf{S} \mathbf{K}_{MM}^{-1} \mathbf{k}_i \tag{8}$$

Note that the complete variational distribution used in SVGP is given by

$$q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f}|\mathbf{u}, \mathbf{X}) q(\mathbf{u}) \tag{9}$$

with a marginal distribution given by

$$q(\mathbf{f}) = \int d\mathbf{u} \; q(\mathbf{f}, \mathbf{u}) = \mathcal{N}(\mathbf{f}|\mu_f(\mathbf{X}), \boldsymbol{\Sigma}_f(\mathbf{X})) \tag{10}$$

[1] Note that here and throughout this work we focus on the regression case.

where $\mathbf{\Sigma}_f(\mathbf{X})$ is the $N \times N$ covariance matrix

$$\mathbf{\Sigma}_f(\mathbf{X}) = \widetilde{\mathbf{K}}_{NN} + \mathbf{K}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{S}\mathbf{K}_{MM}^{-1}\mathbf{K}_{MN} \quad (11)$$

The objective $\mathcal{L}_{\rm svgp}$, which depends on $\boldsymbol{m}, \mathbf{S}, \mathbf{Z}, \sigma_{\rm obs}$ and the various kernel hyperparameters, can then be maximized with gradient methods. Since the expected log likelihood in Eqn. 6 factorizes as a sum over data points $(y_i, \mathbf{x}_i)$ it is amenable to stochastic gradient methods, and thus SVGP can be applied to very large datasets.

2.3 DEEP GAUSSIAN PROCESSES

Deep Gaussian Processes (Damianou and Lawrence, 2013) are a natural generalization of GPs in which a sequence of GP layers form a hierarchical model in which the outputs of one GP layer become the inputs of the subsequent layer, resulting in a flexible, compositional function prior. In the simplest case of a 2-layer DGP with a univariate continuous output y and with a GP layer of width W fed into the topmost GP, the joint likelihood for a dataset $(\mathbf{y}, \mathbf{X})$ is given by[2]

$$p(\mathbf{y}, \mathbf{f}, \mathbf{G}|\mathbf{X}) = p(\mathbf{y}|\mathbf{f}, \sigma_{\rm obs}^2)p(\mathbf{f}|\mathbf{G})p(\mathbf{G}|\mathbf{X}) \quad (12)$$

Here $\mathbf{G}$ is a matrix of latent function values of size $N \times W$, i.e. g_{iw} denotes the latent function value of the $w^{\rm th}$ GP in the first layer evaluated at the $i^{\rm th}$ input $\mathbf{x}_i$. For $i = 1, ..., N$ the vector $\mathbf{g}_i \equiv g_{i,:}$ of dimension W then serves as the input to the second GP denoted by $\mathbf{f}$. See Fig. 1 for an illustration. Throughout this work we assume that the prior over $\mathbf{G}$ factorizes as $p(\mathbf{G}|\mathbf{X}) = \prod_{w=1}^{W} p(\mathbf{g}_w|\mathbf{X})$, with each GP governed by its own kernel. Inference in this model is intractable, necessitating approximate methods. A popular approach is variational inference, which we now briefly review.

2.3.1 DOUBLY STOCHASTIC VARIATIONAL INFERENCE

The stochastic variational inference approach described in Sec. 2.2.1 can be generalized to the DGP setting (Salimbeni and Deisenroth, 2017). Proceeding in analogy to SVGP, we introduce inducing variables and inducing point locations for each GP and form a factorized variational distribution $Q(\mathbf{f}, \mathbf{u}_f, ..., \mathbf{g}_W, \mathbf{u}_{gW})$ with each factor of the form in Eqn. 9. The variational distribution Q can then be used to form the ELBO

$$\mathcal{L}_{\rm dsvi} = \mathbb{E}_Q \left[\log p(\mathbf{y}|\mathbf{f}, \sigma_{\rm obs}^2)\right] - \sum \mathrm{KL} \quad (13)$$

where $\sum$ KL denotes a sum over the various KL divergences for the inducing variables $\{\mathbf{u}_f, ..., \mathbf{u}_{gW}\}$. Crucially, Q is easy to sample from since it suffices to sample from the various marginals $\{q(f_i), ..., q(g_{iw})\}$ and so the expected log likelihood term in Eqn. 13 factorizes[3] into a sum over data points $(y_i, \mathbf{x}_i)$, enabling mini-batch training. As in SVGP, the topmost layer of latent function values $\mathbf{f}$ can be integrated out analytically. All remaining latent variables (namely $\mathbf{G}$) must be sampled, necessitating the use of the reparameterization trick (Price, 1958; Salimans et al., 2013) and resulting in 'doubly' stochastic gradients. For more details we refer the reader to (Salimbeni and Deisenroth, 2017).

2.3.2 DGP PREDICTIVE DISTRIBUTIONS

We review the structure of the posterior predictive distribution that results from the variational inference procedure described in the previous section, as this will serve as motivation for the introduction of Deep Sigma Point Processes in Sec. 3. We first note that in the (single-layer) SVGP case the predictive distribution at input $\mathbf{x}_*$ is given by the Normal distribution

$$p(y_*|\mathbf{x}_*) = \mathcal{N}(y_*|\mu_f(\mathbf{x}_*), \sigma_f(\mathbf{x}_*)^2 + \sigma_{\rm obs}^2) \quad (14)$$

where $\mu_f(\mathbf{x}_*)$ is the predictive mean function in Eqn. 7 and $\sigma_f(\mathbf{x}_*)^2$ is the latent function variance in Eqn. 8. The predictive distribution for the DGP is an immediate generalization of Eqn. 14. In particular in the DGP case the predictive distribution is given by a *continuous mixture* of Normal distributions of the form in Eqn. 14. In more detail, for the 2-layer DGP in Eqn. 12, the predictive distribution is given by

$$\mathbb{E}_{\prod_{w=1}^{W} q(g_{*w}|\mathbf{x}_*)} \left[\mathcal{N}(y_*|\mu_f(\mathbf{g}_*), \sigma_f(\mathbf{g}_*)^2 + \sigma_{\rm obs}^2)\right] \quad (15)$$

Since this expectation is intractable, in practice it must be approximated with Monte Carlo samples, i.e. as a finite mixture of Normal distributions.

2.4 PPGPR

In this section we review a recently introduced approach to scalable GP regression (PPGPR; Jankowiak et al. (2020)), as it will serve as motivation for Deep Sigma Point Processes in Sec. 3. In PPGPR one takes the family of predictive distribution in Eqn. 14—parameterized by $\boldsymbol{m}$, $\mathbf{S}$, $\mathbf{Z}$, and the kernel hyperparameters—as the model class, and fits the model using gradient-based maximum likelihood estimation. Jankowiak et al. (2020) argue that this class of models achieves good performance because the training objective directly targets the distribution used to make predictions at test time. In more detail the PPGPR

[2] We limit our discussion to this case to simplify notation; generalizations to multiple outputs and multiple layers are straightforward.

[3] Since e.g. $\mathbf{\Sigma}_f(\mathbf{X})_{ii} = \sigma_f(\mathbf{x}_i)^2$ only depends on $\mathbf{x}_i$.

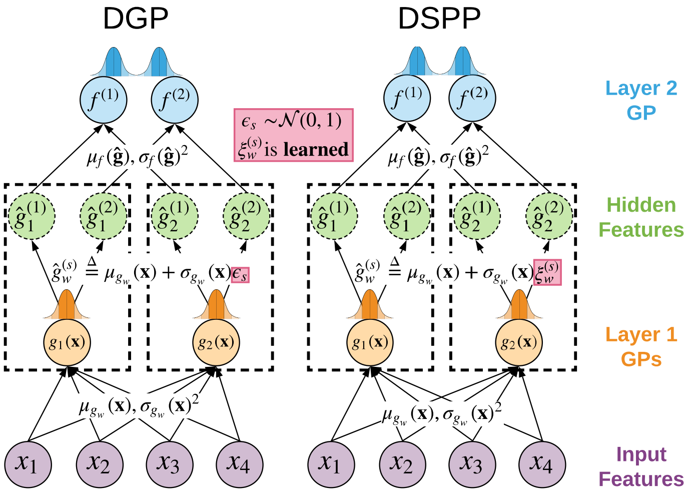
DGP
DSPP
$f^{(1)}$
$f^{(2)}$
$\epsilon_s \sim \mathcal{N}(0,1)$
$\xi_w^{(s)}$ is learned
$\mu_f(\hat{\mathbf{g}}), \sigma_f(\hat{\mathbf{g}})^2$
$\hat{g}_1^{(1)}$
$\hat{g}_1^{(2)}$
$\hat{g}_2^{(1)}$
$\hat{g}_2^{(2)}$
$\hat{g}_w^{(s)} \triangleq \mu_{g_w}(\mathbf{x}) + \sigma_{g_w}(\mathbf{x})\epsilon_s$
$\hat{g}_w^{(s)} \triangleq \mu_{g_w}(\mathbf{x}) + \sigma_{g_w}(\mathbf{x})\xi_w^{(s)}$
$g_1(\mathbf{x})$
$g_2(\mathbf{x})$
$\mu_{g_w}(\mathbf{x}), \sigma_{g_w}(\mathbf{x})^2$
x_1
x_2
x_3
x_4
Layer 2 GP
Hidden Features
Layer 1 GPs
Input Features

Figure 1: We depict the computional flow of a 2-layer DGP (left; see Sec. 2.3) and a 2-layer DSPP (right; see Sec. 3). Input features $\mathbf{x}$ are fed through a set of W (here $W = 2$) Gaussian Processes by computing $\mu_{g_w}(\mathbf{x})$ and $\sigma_{g_w}(\mathbf{x})$ (Eqns. 7 and 8). Hidden features for the next GP layer (f) are computed from each $\mu_{g_w}(\mathbf{x})$ and $\sigma_{g_w}(\mathbf{x})$ by sampling from $\mathcal{N}(\mu_{g_w}(\mathbf{x}), \sigma_{g_w}(\mathbf{x}))$ in the DGP case, or by applying one of the quadrature rules discussed in Sec. 3.2 (here QR3), with the $\xi_w^{(s)}$ treated as learnable parameters. The final predictive distribution is a mixture of S Gaussians, where each Gaussian depends on one of the S sampled feature sets in the DGP case, or on one of the S deterministic quadrature-dependent feature sets in the DSPP case. The DGP is trained by gradient descent on $\mathcal{L}_{\text{dsvi}}$ (Eqn. 13), and the DSPP is trained via $\mathcal{L}_{\text{dspp}}$ (Eqn. 21). See Sec. 3.3 for more discussion on the relationship between DGPs and DSPPs.

objective is given by

$$\begin{aligned}\mathcal{L}_{\text{ppgpr}} &= \sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i) - \beta_{\text{reg}}\text{KL}(q(\mathbf{u})|p(\mathbf{u})) \\ &= \sum_{i=1}^{N} \log \mathcal{N}(y_i|\mu_f(\mathbf{x}_i), \sigma_f(\mathbf{x}_i)^2 + \sigma_{\text{obs}}^2) \\ &\quad - \beta_{\text{reg}}\text{KL}(q(\mathbf{u})|p(\mathbf{u}))\end{aligned} \tag{16}$$

where $\beta_{\text{reg}} > 0$ is a hyperparameter and $\text{KL}(q(\mathbf{u})|p(\mathbf{u}))$ serves as a regularizer. Note that the objective in Eqn. 16 looks deceptively similar to the SVGP objective in Eqn. 6; the crucial difference is where $\sigma_f(\mathbf{x}_i)^2$ appears. This difference is a result of the fact that the PPGPR objective directly targets the predictive distribution, while the SVGP objective targets minimizing a KL divergence between the variational distribution and the exact posterior. In SVGP the posterior predictive is then formed in a second step and does not directly enter the training procedure. PPGPR will serve as one of our baselines in Sec. 5.

3 DEEP SIGMA POINT PROCESSES

We now describe the class of models that is the focus of this work. Our approach is motivated by the good empirical performance exhibited by PPGPR, a class of parametric GP models explored in (Jankowiak et al., 2020) and reviewed in Sec. 2.4. Crucially, this approach to scalable GP regression relies on an objective function that is formulated in terms of the predictive distribution. We would like to apply this approach to the DGP setting but, unfortunately, the form of the predictive distribution for DGPs (see Eqn. 15) presents an immediate obstacle: Eqn. 15 is a continuous mixture of Normal distributions and cannot be computed in closed form. In other words, the analog of the PPGPR objective in Eqn. 16 would involve the logarithm of the expectation in Eqn. 15. While this quantity could be approximated with a Monte Carlo estimator, because of the outer logarithm the result would be a biased estimator. Instead of trying to address this obstacle directly and construct an unbiased (gradient) estimator, we instead adopt a simpler solution: we replace the continuous mixture with a (parametric) *finite* mixture.[4]

To define a finite parametric family of mixture distributions, we essentially apply a sigma point approximation or quadrature-like integration rule to Eqn. 15. To make this concrete, suppose the width of the first GP layer in Eqn. 15 is $W = 2$. Then for an input $\mathbf{x}_i$ we can write

$$p_{\text{dspp}}(y_i|\mathbf{x}_i) = \int d\mathbf{g}_i \mathcal{N}(y_i|\mu_f(\mathbf{g}_i), \sigma_f(\mathbf{g}_i)^2 + \sigma_{\text{obs}}^2) \prod_{w=1}^{2} q(g_{iw}|\mathbf{x}_i) \tag{17}$$

The simplest ansatz for converting Eqn. 17 into a finite mixture uses Gauss-Hermite quadrature, i.e. we approximate $q(g_{i1}|\mathbf{x}_i)$ with a S-component mixture of Dirac delta distributions controlled by weights $\omega_1^{(s)}$ and quadra-

[4] In Sec. C we report empirical results comparing to the 'direct' (biased) approach in which the predictive distribution is given as a continuous mixture. We find that this model variant is outperformed by the DSPP, presumably because of the additional model flexibility that results from the learned quadrature rules we adopt in Sec. 3.2.

ture points $\xi_1^{(s)}$

$$q(g_{i1}|\mathbf{x}_i) = \sum_{s_1=1}^{S} \omega_1^{(s_1)} \delta\left(g_{i1} - \left(\mu_{g_1}(\mathbf{x}_i) + \xi_1^{(s_1)} \sigma_{g_1}(\mathbf{x}_i)\right)\right) \tag{18}$$

with an analogous ansatz for $q(g_{i2}|\mathbf{x}_i)$. For example for $S = 3$ we would have

$$\begin{aligned} \xi^{(1)} &= -\sqrt{3} \qquad \xi^{(2)} = 0 \qquad \xi^{(3)} = -\sqrt{3} \\ \omega^{(1)} &= \frac{1}{6} \qquad \omega^{(2)} = \frac{2}{3} \qquad \omega^{(3)} = \frac{1}{6} \end{aligned} \tag{19}$$

for both g_{i1} and g_{i2}. Making these replacements in Eqn. 17 then yields a mixture with S^2 components:

$$\begin{aligned} p_{\text{dspp}}(y_i|\mathbf{x}_i) &= \sum_{s_1}^{S} \sum_{s_2}^{S} \omega_1^{(s_1)} \omega_2^{(s_2)} \times \qquad (20) \\ \mathcal{N}(y_i | \mu_f(\mu_{g_1}(\mathbf{x}_i) &+ \xi_1^{(s_1)} \sigma_{g_1}(\mathbf{x}_i), \mu_{g_2}(\mathbf{x}_i) + \xi_1^{(s_2)} \sigma_{g_2}(\mathbf{x}_i)), \\ \sigma_f(\mu_{g_1}(\mathbf{x}_i) &+ \xi_1^{(s_1)} \sigma_{g_1}(\mathbf{x}_i), \mu_{g_2}(\mathbf{x}_i) + \xi_1^{(s_2)} \sigma_{g_2}(\mathbf{x}_i))) \end{aligned}$$

Thus for a 2-layer model where the first GP layer has width W this particular quadrature rule leads to a predictive distribution that is a mixture of S^W Normal distributions, each of which is parameterized by compositition of mean and variance functions of the form in Eqn. 7 and Eqn. 8. This exponential growth in the number of mixture components is potentially problematic; we defer a more detailed discussion of alternative—in particular more compact—quadrature rules to Sec. 3.2.

3.1 TRAINING OBJECTIVE

Now that we have defined the class of parametric regression models we are interested in, we can define our training objective. As in Jankowiak et al. (2020), we define an objective function that corresponds to regularized maximum likelihood estimation

$$\mathcal{L}_{\text{dspp}} = \sum_{i=1}^{N} \log p_{\text{dspp}}(y_i|\mathbf{x}_i) - \beta_{\text{reg}} \sum \text{KL} \tag{21}$$

where $\beta_{\text{reg}} > 0$ is an optional regularization constant and $\sum$ KL is a sum over KL divergences of inducing variables (one for each GP) just as in the DGP objective, Eqn. 13. Just as in 'Doubly Stochastic Variational Inference' for DGPs (Sec. 2.3.1), this objective—which depends on σ_{obs} as well as parameters $\boldsymbol{m}, \mathbf{S}, \mathbf{Z}$ and various kernel hyperparameters for each GP—can be optimized using stochastic gradient methods. In contrast to DGPs, this optimization is only 'singly stochastic,' i.e. we only subsample data points and not latent function values.

3.2 QUADRATURE RULES

The Gauss-Hermite quadrature rule used to motivate DSPPs in Eqn. 18 is intuitive, but is of limited practical use, since it leads to an exponential blow-up in the number of mixture components used to define the DSPP. It is therefore essential to consider alternative quadrature rules that are more compact. We emphasize at the outset that our goal is *not* to accurately estimate the intractable expectation in Eqn. 17. Rather, our goal is to construct a flexible family of parametric distributions that are governed by well-behaved mean and variance functions that benefit from compositional structure. Consequently, there is no need to restrict ourselves to quadrature rules derived from gaussian integrals—indeed empirically we find that the quadrature rule in Eqn. 18 is too rigid. We now describe three more flexible alternatives.

QR1 In the first quadrature rule we choose quadrature points that follow the same factorized structure that is evident in the double sum in Eqn. 20, i.e. we still use a grid of S^W quadrature points for a 2-layer DSPP where the first GP layer has width W. That is we make the substitution

$$\begin{aligned} &\prod_{w=1}^{W} q(g_{iw}|\mathbf{x}_i) \to \qquad (22) \\ &\sum_{\boldsymbol{s}} \omega^{(\boldsymbol{s})} \prod_{w=1}^{W} \delta\left(g_{iw} - \left(\mu_{g_w}(\mathbf{x}_i) + \xi_w^{(s_w)} \sigma_{g_w}(\mathbf{x}_i)\right)\right) \end{aligned}$$

where $\boldsymbol{s} = (s_1, ..., s_W) \in \{1, ..., S\}^W$ is a multi-index. In contrast to the Gauss-Hermite quadrature rule, however, we choose the quadrature points $\{\xi_w^{(s_w)}\}$ to be learnable parameters (for a total of $S \times W$ real parameters). In addition we replace the Gauss-Hermite weights for each mixture—which are given as a product over the W GPs in the first layer, i.e. $\prod_w \omega_w^{(s_w)}$ for each multi-index $\boldsymbol{s}$—by S^W-many learnable parameters that sum to unity $\{\omega^{(\boldsymbol{s})}\}$.

QR2 The second rule is identical to QR1 except the quadrature points are forced to be symmetric, i.e. $\xi_w^{(s_w)} = -\xi_w^{(S+1-s_w)}$ for $s_w = 1, ..., S$ and $w = 1, ..., W$.

QR3 In the third quadrature rule we abandon the factorized structure of QR1 and QR2, thus liberating us from the exponential growth of mixture components. To do this we effectively 'line-up' the quadrature points across the different GPs $g_1, ..., g_W$, making the substitution

$$\begin{aligned} &\prod_{w=1}^{W} q(g_{iw}|\mathbf{x}_i) \to \qquad (23) \\ &\sum_{s=1}^{S} \omega^{(s)} \prod_{w=1}^{W} \delta\left(g_{iw} - \left(\mu_{g_w}(\mathbf{x}_i) + \xi_w^{(s)} \sigma_{g_w}(\mathbf{x}_i)\right)\right) \end{aligned}$$

where there is a set of S learnable quadrature weights $\{\omega^{(s)}\}_{s=1}^{S}$ and S learnable quadrature points $\{\xi_w^{(s)}\}_{s=1}^{S}$ defined by $S \times W$ real parameters. Here S is a parameter that we control; in particular it has no relationship to W and can be as small as $S = 1$. We explore the empirical performance of these quadrature rules in Sec. 5.1.

3.3 DISCUSSION

We use this section to clarify the relationship between variational DGPs and DSPPs. DSPPs differ from DGPs in two important respects (also see Fig. 1):

1. **Objective function:** The DGP is trained with an ELBO (Eqn. 13) and the DSPP is trained via a regularized maximum likelihood objective (Eqn. 21) that directly targets the DSPP predictive distribution.

2. **Treatment of latent function values:** In the DGP latent function values not at the top of the hierarchy (e.g. $\mathbf{G}$ in Eqn. 12) are *sampled* while in the DSPP they are *parameterized* via a learnable quadrature rule as in Eqn. 23.

Indeed this latter point is made explicit by our quadrature rules—see Eqn. 22 and Eqn. 23—which should be compared to the reparameterization trick used during DGP training, which implicitly makes the substitution

$$q(g_{iw}|\mathbf{x}_i) \to \mathbb{E}_{\mathcal{N}(\epsilon|0,1)}\left[\delta\left(g_{iw} - (\mu_{g_w}(\mathbf{x}_i) + \epsilon\sigma_{g_w}(\mathbf{x}_i))\right)\right]$$

and approximates the expectation using Monte Carlo samples $\{\epsilon_s\}$ with $\epsilon_s \sim \mathcal{N}(\cdot|0,1)$.

Note that in other respects variational DGPs and DSPPs are very similar. For example, apart from the parameters defining the learned quadrature rule in a DSPP, they make use of the same parameters. Similarly, their predictive distributions have similar forms, with the difference that the DSPP utilizes a finite mixture of Normal distributions, while the DGP predictive distribution is a continuous mixture of Normal distributions.

4 RELATED WORK

We discuss some work related to DSSPs, noting that we review some of the relevant literature in Sec. 2. The use of pseudo-inputs and inducing point methods to scale-up Gaussian Process inference has spawned a large literature, especially in the context of variational inference (Snelson and Ghahramani, 2006; Titsias, 2009; Hensman et al., 2013; Cheng and Boots, 2017). While variational inference remains the most popular inference algorithm for scalable GPs, researchers have also explored different variants of Expectation Propagation (Hernández-Lobato and Hernández-Lobato, 2016) as well as Stochastic gradient Hamiltonian Monte Carlo (Havasi et al., 2018).

Table 1: Average ranking of different quadrature rules (further to the left is better). CRPS is the Continuous Ranked Probability Score, a popular calibration metric for regression (Gneiting and Raftery, 2007). Rankings are aggregated across the smallest 8 UCI datasets and train/test/validation splits. See Sec. 5.1 for details.

	DSPP-QR1	DSPP-QR2	DSPP-QR3
NLL	2.01	2.44	**1.37**
RMSE	1.79	2.27	**1.62**
CRPS	1.73	2.51	**1.51**

Table 2: Average ranking of methods in Sec. 5.2 (lower is better). Rankings are aggregated across all 120 pairs of dataset and train/test/validation split.

	OD-SVGP	PPGPR	DGP	γ-DGP	DSPP
NLL	4.38	2.33	3.55	3.60	**1.15**
RMSE	3.28	2.85	2.98	3.32	**2.58**
CRPS	4.42	2.47	3.66	2.93	**1.52**

Deep Gaussian Processes were introduced in (Damianou and Lawrence, 2013), with recent approaches to variational inference for DGPs described by Salimbeni and Deisenroth (2017). Cutajar et al. (2017) introduce a hybrid model formulated with random feature expansions that combines features of DGPs and neural networks. This class of models bears some resemblance to DSPPs; this is especially true for the `VAR-FIXED` variant, in which spectral frequencies are treated deterministically. Importantly, since Cutajar et al. (2017) rely on variational inference, their training objective does not directly target the predictive distribution. As we show empirically in Sec. 5, the mismatch between the training objective and test time predictive distributions for variational DGPs severely degrades predictive performance; we expect this is also true of the approach in (Cutajar et al., 2017).

DSPPs can also be motivated by Direct Loss Minimization, which emerges from a view of approximate inference as regularized loss minimization (Sheth and Khardon, 2017). This connection is somewhat loose, however, since our fully parametric models dispose of approximate posterior (or quasi-posterior) distributions entirely.

5 EXPERIMENTS

In this section we explore the empirical performance of the Deep Sigma Point Process introduced in Sec. 3. Throughout DSPPs use diagonal (i.e. mean field) covari-

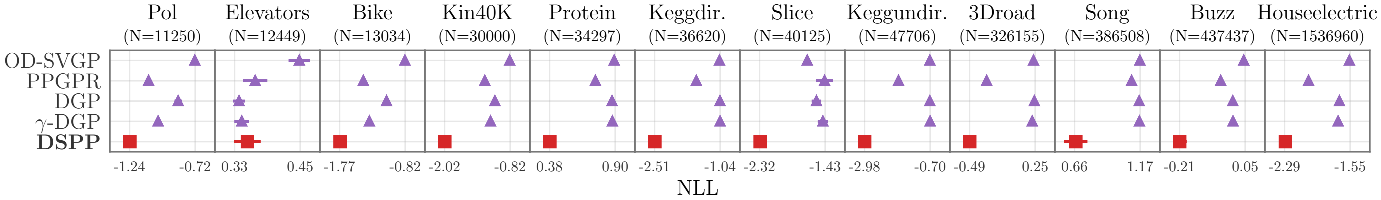

Figure 2: We depict negative log likelihoods (NLLs) for the 12 univariate regression datasets in Sec. 5.2 (further to the left is better). Results are averaged over ten random train/test/validation splits. Here and throughout uncertainty bars depict standard errors.

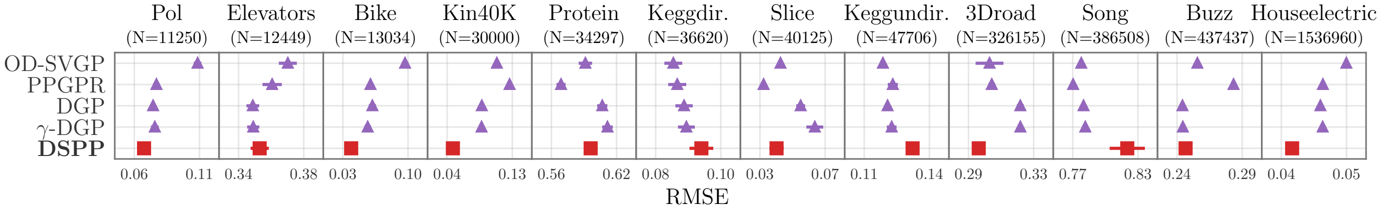

Figure 3: We depict root mean squared errors (RMSEs) for the 12 univariate regression datasets in Sec. 5.2 (further to the left is better). Results are averaged over ten random train/test/validation splits.

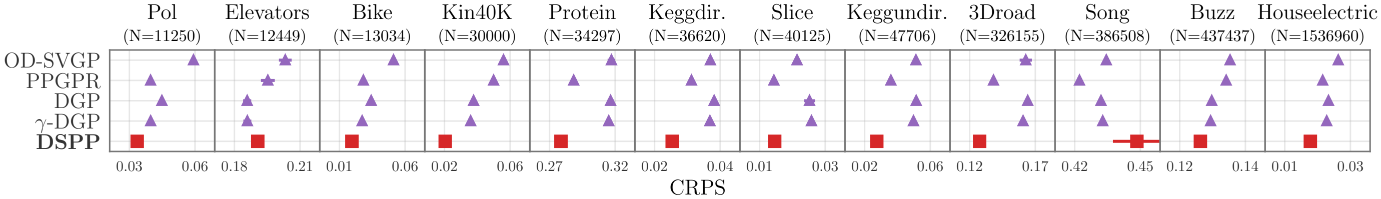

Figure 4: We depict the Continuous Ranked Probabilistic Score (CRPS; Gneiting and Raftery (2007)) for the 12 univariate regression datasets in Sec. 5.2 (further to the left is better). CRPS is a popular calibration metric for regression. Results are averaged over ten random train/test/validation splits.

ance matrices $\mathbf{S}$. Except for Sec. 5.4 where we consider 3-layer models, all DGPs and DSPPs considered in our experiments have two layers. For details about kernels, mean functions, layer widths, numbers of inducing points, etc., refer to Sec. A in the appendix.

5.1 QUADRATURE RULE ABLATION STUDY

We begin by exploring the impact of the three different quadrature rules defined in Sec. 3.2. To do so we train 2-layer DSPPs on the 8 smallest univariate regression datasets described in the next section. In particular we compare QR1 and QR2 with[5] $S = 3$ to QR3 with $S = 10$. The results are summarized in Table 1, with more detailed results to be found in Sec. C in the appendix. The upshot is that the three quadrature rules give largely comparable performance, with some preference for the most flexible quadrature rule, QR3. Since this quadrature rule avoids exponentially many quadrature points—and the computational cost can be carefully controlled by choosing S appropriately—we use QR3 in the remainder of our experiments. Indeed this choice is essential for training 3-layer DSPPs in Sec. 5.4, which would otherwise be too computationally expensive.

5.2 UNIVARIATE REGRESSION

We reproduce a univariate regression experiment described in (Jankowiak et al., 2020). In particular we consider consider twelve datasets from the UCI repository (Dua and Graff, 2017), with the number of data points in the range $10^4 \lessapprox N \lessapprox 10^6$ and the number of input dimensions in the range $3 \le \mathrm{Dim}(\mathbf{x}) \le 380$.

We consider four strong GP baselines—two single-layer models: (**OD-SVGP**) the orthogonal basis decoupling method described in Cheng and Boots (2017) and Salimbeni et al. (2018); (**PPGPR**) the Parametric Predictive GP regression method described in Sec. 2.4; as well as two 2-layer models: (**DGP**) a variational DGP as described in Sec. 2.3; and (**γ-DGP**) the robust DGP described in

[5] Since we consider $W \in \{3, 4\}$ this corresponds to $S^W \in \{27, 81\}$ mixture components.

Table 3: Average ranking of methods in Sec. 5.3 (lower is better). Rankings are aggregated across all 25 pairs of dataset and train/test/validation split.

	SVGP	PPGPR	DGP	γ-DGP	DSPP
NLL	4.16	2.20	4.24	3.40	**1.00**
MRMSE	2.52	4.72	**2.04**	3.60	2.12

Knoblauch (2019); Knoblauch et al. (2019).[6] Results for OD-SVGP and PPGPR are reproduced from (Jankowiak et al., 2020).

Our results are summarized in Figs. 2-4 and Table 2. We find that in aggregate the DSPP outperforms all the baselines in terms of log likelihood, RMSE, and CRPS (also see Table 6 in Sec. C in the appendix). In particular, averaging across all twelve datasets, the DSPP outperforms the DGP by ~ 0.75 nats and the PPGPR by ~ 0.47 nats w.r.t. log likelihood. Strikingly, the second strongest baseline is the (single-layer) PPGPR described in Sec. 2.4. The fact that the PPGPR is able to outperform a 2-layer DGP highlights the advantages of a training procedure that directly targets the predictive distribution. Since the DSPP is in effect a much more flexible version of the PPGPR, it yields even better predictive performance, especially with respect to log likelihood. Indeed while the DGP achieves good RMSE performance on most datasets, posterior approximations degrade the calibration of the test time predictive distribution (as measured by log likelihood and CRPS).

5.3 MULTIVARIATE REGRESSION

We conduct a multivariate regression experiment using five robotics datasets, two of which were collected from real-world robots and three of which were generated using the MuJoCo physics simulator (Todorov et al., 2012). In all five datasets the input and output dimensions correspond to various joint positions/velocities/etc. of the robot, with the number of data points in the range $10^4 \lessapprox N \lessapprox 10^5$, the number of input dimensions in the range $10 \leq \mathrm{Dim}(\mathbf{x}) \leq 23$, and the number of output dimensions in the range $7 \leq \mathrm{Dim}(\mathbf{y}) \leq 10$. These datasets have been used in a number of papers, including (Vijayakumar and Schaal, 2000; Cheng and Boots, 2017).

All of our models follow the structure of the 'linear model of coregionalization' (Alvarez et al., 2012), specifically along the lines of 'Semiparametric latent factor models' (Seeger et al., 2005); for more details see Sec. A.3 in the supplementary materials.

We consider four strong GP baselines. In particular we consider two single-layer models: (**SVGP**) as described in Sec. 2.2.1 and (Hensman et al., 2013); (**PPGPR**) the Parametric Predictive GP regression method described in Sec. 2.4; as well as two 2-layer models: (**DGP**) a variational DGP as described in Sec. 2.3; and (**γ-DGP**) the robust DGP described in Knoblauch (2019); Knoblauch et al. (2019).

Our results are summarized in Fig. 5 and Table 3; see Sec. C in the appendix for additional results. We find that the DSPP outperforms all the baselines w.r.t. NLL, and achieves comparable MRMSE performance to the DGP. Averaged across all five datasets, the DSPP outperforms the DGP by ~ 0.82 nats and PPGPR by ~ 0.17 nats w.r.t. log likelihood. Note that while the PPGPR achieves good performance on log likelihoods, its MRMSE performance is substantially worse than the DSPP.

5.4 MULTILAYER MODELS

In the above experiments we have limited ourselves to 2-layer DSPPs. Here we investigate the predictive performance of 3-layer models, in particular comparing to 3-layer DGPs. Our results for four univariate regression datasets are summarized in Fig. 6. For both DGPs and DSPPs we find that, depending on the dataset, adding a third layer can improve NLLs, but that these gains are somewhat marginal compared to the gains from moving from single-layer to two-layer models. DSPPs exhibit the best log likelihoods, while RMSE results are somewhat more mixed, with the DSPP and PPGPR obtaining the lowest RMSE depending on the dataset.

5.5 COMPARISON WITH DEEP KERNEL LEARNING

From the previous sets of results we see that DSPP models tend to outperform their single-layer counterparts, both in terms of NLL and RMSE. A natural question is whether or not these performance improvements can be obtained with other classes of hierarchical models. More specifically, could we achieve similar results if the hierarchical features are extracted using traditional neural networks rather than GPs? To answer this question, we consider deep kernel learning (DKL) variants of SVGP, PPRPR, DPG, and DSPP in which the input features to each model are extracted by a neural network (Calandra et al., 2016; Wilson et al., 2016). The neural network parameters are optimized end-to-end alongside the GP/DGP/DSPP parameters. We conduct experiments on 4 medium-sized UCI regression datasets. For all DKL models we use a

[6] This robust DGP can be viewed as a variant of the DGP described in Sec. 2.3 in which the inference procedure is modified by replacing the expected log likelihood loss with a gamma divergence in which the likelihood is raised to a power: $\log p(\mathbf{y}|\mathbf{f}) \to p(\mathbf{y}|\mathbf{f})^{\gamma-1}$.

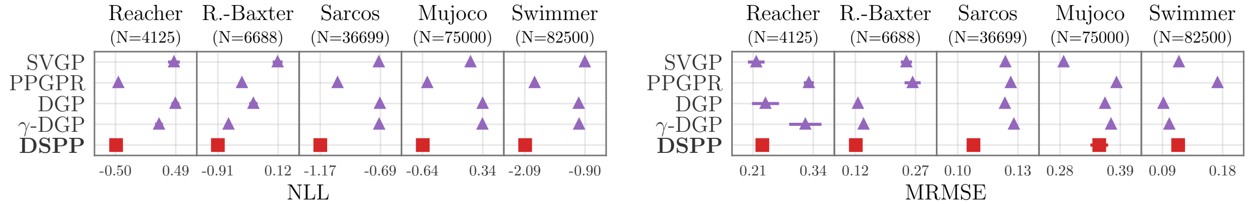

Figure 5: We depict NLLs (left) and mean root mean squared errors (MRMSEs, right)—i.e. RMSEs averaged across all output dimensions—for the 5 multivariate regression datasets in Sec. 5.3 (lower is better). Results are averaged over five random train/test/validation splits. Note that NLLs are normalized by the number of output dimensions.

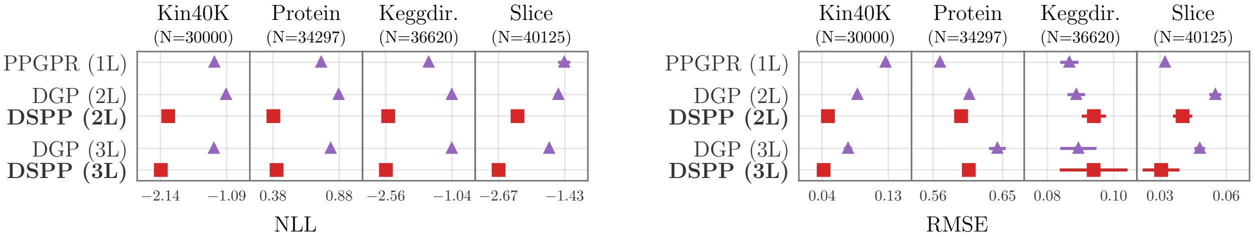

Figure 6: We depict NLLs (left) and RMSEs (right) for the multi-layer experiment in Sec. 5.4 (lower is better). Results for 1L and 2L (respectively, 3L) models are averaged over ten (respectively, five) random train/test/validation splits.

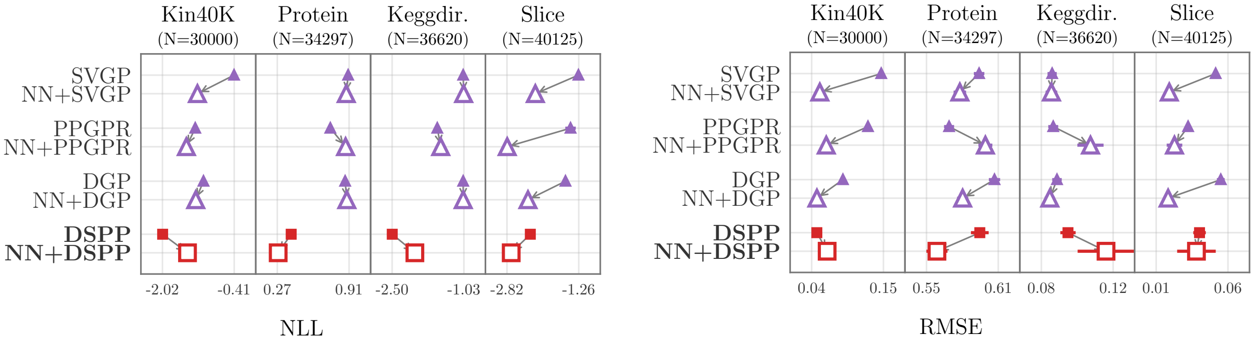

Figure 7: We compare NLLs (left) and RMSEs (right) of neural-network modulated (deep kernel learning) variants of GP/DGP/DSPP models (lower is better). Results are presented for 4 of the UCI regression datasets in Sec. 5.2, averaged over ten random train/test/validation splits. See Sec. 5.5 for details.

5-layer neural network architecture proposed by Wilson et al. (2016) to extract features (see Sec. A.5 for details).

In Fig. 7 we compare the neural network modulated models (denoted by "NN+") with their standard counterparts. We find that adding a neural network to DSPP models has limited impact on performance, improving on some datasets and not on others. For SVGP/PPGPR/DGP models, neural networks improve RMSE but have mixed effects on NLL. On three of the four datasets, none of the NN + SVGP/PPGPR/DGP models matches the NLL of the standard DSPP. This is particularly notable for the NN+PPGPR model, as it only differs from the standard DSPP model in terms of its "feature extractor" layer (neural networks versus GP/quadrature). These results suggests that hidden GP layers in DSPPs extract features that are complementary to those extracted by neural networks, while enjoying favorable regularization properties.

6 DISCUSSION

We motivated Deep Sigma Point Processes as a finite family of parametric distributions whose structure mirrors the DGP predictive distribution in Eqn. 15. It would be interesting to consider model variants that retain a continuous mixture distribution as in Eqn. 15 while leveraging the flexibility inherent in the learned quadrature rules described in Sec. 3.2. Another open question is whether DSPPs can be fruitfully applied to other likelihoods, for example those that arise in classification. We leave the exploration of these directions to future work.

References

Mauricio A Alvarez, Lorenzo Rosasco, Neil D Lawrence, et al. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3): 195–266, 2012.

Roberto Calandra, Jan Peters, Carl Edward Rasmussen, and Marc Peter Deisenroth. Manifold gaussian processes for regression. In *International Joint Conference on Neural Networks*, 2016.

Ching-An Cheng and Byron Boots. Variational inference for gaussian process models with linear complexity. In *Advances in Neural Information Processing Systems*, 2017.

Kurt Cutajar, Edwin V Bonilla, Pietro Michiardi, and Maurizio Filippone. Random feature expansions for deep gaussian processes. In *International Conference on Machine Learning*, 2017.

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

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Marton Havasi, José Miguel Hernández-Lobato, and Juan José Murillo-Fuentes. Inference in deep gaussian processes using stochastic gradient hamiltonian monte carlo. In *Advances in Neural Information Processing Systems*, 2018.

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

Daniel Hernández-Lobato and José Miguel Hernández-Lobato. Scalable gaussian process classification via expectation propagation. In *Artificial Intelligence and Statistics*, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

Martin Jankowiak, Geoff Pleiss, and Jacob R Gardner. Parametric gaussian process regressors. In *International Conference on Machine Learning*, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learned Representations*, 2015.

Jeremias Knoblauch. Robust deep gaussian processes. *arXiv preprint arXiv:1904.02303*, 2019.

Jeremias Knoblauch, Jack Jewson, and Theodoros Damoulas. Generalized variational inference. *arXiv preprint arXiv:1904.02063*, 2019.

Robert Price. A useful theorem for nonlinear devices having gaussian inputs. *IRE Transactions on Information Theory*, 4(2):69–72, 1958.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

Tim Salimans, David A Knowles, et al. Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):837–882, 2013.

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

Hugh Salimbeni, Ching-An Cheng, Byron Boots, and Marc Deisenroth. Orthogonally decoupled variational gaussian processes. In *Advances in neural information processing systems*, 2018.

Matthias Seeger, Yee-Whye Teh, and Michael Jordan. Semiparametric latent factor models. Technical report, 2005.

Rishit Sheth and Roni Khardon. Excess risk bounds for the bayes risk using variational inference in latent gaussian models. In *Advances in Neural Information Processing Systems*, 2017.

Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, 2006.

Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics*, 2009.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems*, 2012.

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

Andrew G Wilson, Zhiting Hu, Russ R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, 2016.

A EXPERIMENTAL DETAILS

We use Matérn kernels with independent length scales for each input dimension throughout. Throughout we discard input or output dimensions that have negligible variance.

A.1 QUADRATURE RULE ABLATION STUDY

The experimental procedure for this experiment follows that described in the next section, with the difference that the quadrature rule ablation study described in Sec. 5.1 only makes use of the 8 smallest UCI datasets and we consider $W \in \{3, 4\}$.

A.2 UNIVARIATE REGRESSION

We follow the experimental procedure outlined in (Jankowiak et al., 2020). In particular we use the Adam optimizer for optimization with an initial learning rate of $\ell = 0.01$ that is progressively decreased during the course of training (Kingma and Ba, 2015). We use a mini-batch size of $B = 2000$ for the Buzz, Song, 3droad and Houseelectric datasets and $B = 10^3$ for all other datasets. We train for 400 epochs except for the Houseelectric dataset where we train for 150 epochs and the Buzz, Song, and 3droad datasets where we train for 250 epochs. We do 10 train/test/validation splits on all datasets, always in the proportion 15:3:2. All datasets are standardized in both input and output space. For all 2-layer models we use $M = 300$ inducing points for each GP; we initialize with kmeans. For the DSPP we use quadrature rule QR3 with S=10, while for the DGP and γ-DGP we use 10 Monte Carlo samples to approximate the ELBO training objective. We use the validation set to determine a small set of hyperparameters. In particular for the DGP we search over $\beta_{\text{reg}} \in \{0.1, 0.3, 0.5, 1.0\}$ (where, as elsewhere, β_{reg} is a constant that scales the KL regularization term). For the γ-DGP we search over $\gamma \in \{1.01, 1.03, 1.05, 1.1\}$ (with $\beta_{\text{reg}} = 1$). For the DSPP we search over $\beta_{\text{reg}} \in \{0.01, 0.05, 0.2, 1.0\}$. For all 2-layer models we also search over the hyperparameter $W \in \{3, 5\}$, which controls the width of the first layer. For all 2-layer models the mean function in the first layer is linear with learned weights, and the mean function in the second (final) layer is constant with a learned mean. As mentioned in the main text, for the DSPP we use diagonal covariance matrices $\mathbf{S}$ to define variance functions. In contrast, for the DGP and γ-DGP we use full rank covariance matrices parameterized by Cholesky factors, as in the original references. To evaluate log likelihoods for the DGP and γ-DGP we use a Monte Carlo estimator with 32 samples. For all models we do multiple restarts (3) and only train the best initialization to completion (as judged by training NLL); this makes the optimization more robust. As mentioned in the main text, results for PPGPR and OD-SVGP are taken from (Jankowiak et al., 2020).

A.3 MULTIVARIATE REGRESSION

We first describe the linear model of coregionalization (LMC) model structure used by all our multivariate models. We focus on the topmost layer of GPs of width W', as the deeper layers (here only $\mathbf{G}$, since this is a 2-layer model) are structured identically as in the univariate case. Our models are specified by the generative process

$$\begin{aligned} &\mathbf{G} \sim p(\cdot|\mathbf{X}) && \text{[GP prior for first layer]} \\ &\mathbf{F} \sim p(\cdot|\mathbf{G}) && \text{[GP prior for second layer]} \\ &\mathbf{Y} \sim p(\cdot|\mathbf{FA}, \mathbf{\Sigma}_{\text{obs}}) && \text{[likelihood]} \end{aligned} \tag{24}$$

where $\mathbf{F}$ represents a $N \times W'$-dimensional matrix of GP latent function values, $\mathbf{Y}$ is a $N \times D_Y$-dimensional matrix of outputs and $\mathbf{X}$ is the set of D_X-dimensional inputs. Here $\mathbf{A}$ is a $W' \times D_Y$ matrix of learned coefficients that mixes the topmost GPs. The likelihood $p(\mathbf{Y}|\cdot)$ is a Normal likelihood with a (block-)diagonal covariance matrix $\mathbf{\Sigma}_{\text{obs}}$ specified by D_Y learnable parameters. Throughout our experiments we choose $W' = D_Y$ and treat W (the width of the first GP layer) as a hyperparameter.

Our experimental procedure for the multivariate regression experiments follows that of the univariate regression experiments described in the previous section, with the following differences. For the DSPP we use $S = 8$ quadrature points. For the 2-layer models we use $M = 150$ inducing points, while for the single-layer models we use $M = 300$ inducing points. We use a mini-batch size of $B = 400$ for all datasets and train for 300 epochs. To evaluate log likelihoods for the DGP and γ-DGP we use a Monte Carlo estimator with 16 samples. We use 5 random train/test/validation splits for each dataset.

A.4 MULTILAYER MODELS

The experimental procedure used for the multilayer experiments in Sec. 5.4 follows the procedure described in Sec. A.2, with the following differences. For 3-layer models we use 5 instead of 10 random train/test/validation splits. In addition to searching over β_{reg} and the layer width W[7] for 3-layer models we also search over several discrete topology choices. In particular the first layer always uses a linear mean function and the final layer always uses a constant mean function, but the structure of the second layer—in particular how it depends on the outputs of the first layer—differs:

[7] Note that here W is the width of the first as well as the second layer.

1. the 2^{nd} layer mean function is linear and depends on the outputs of the 1^{st} layer (i.e. vanilla feedforward)
2. the 2^{nd} layer mean function is linear and depends on the inputs $\mathbf{x}$ (but the kernel function only depends on the outputs of the 1^{st} layer)
3. the 2^{nd} layer mean function is linear and depends on the outputs of the 1^{st} layer *and* the inputs $\mathbf{x}$ (but the kernel function only depends on the outputs of the 1^{st} layer)
4. the 2^{nd} layer mean function is linear and both the mean function and kernel function depend on the inputs $\mathbf{x}$ and the outputs of the 1^{st} layer

Note that this search over topologies applies to both DSPPs and DGPs.

A.5 DEEP KERNEL LEARNING REGRESSION

The neural network variants of SVGP / PPGPR / DGP / DSPP all use the same 5-layer feature extractor proposed by Wilson et al. (2016). The layers have 1000, 1000, 500, 50, and 20 hidden units (respectively). The inputs to the SVGP / PPGPR / DGP / DSPP models are the 20-dimensional extracted features. We apply batch normalization (Ioffe and Szegedy, 2015) and a ReLU non-linearity after the first four layers. We z-score the final set of extracted features (out of the $d = 20$ layer), which we accomplish using a batch normalization layer without any learned affine transformation. The neural network parameters are trained jointly with the SVGP / PPGPR / DGP / DSPP parameters using the Adam optimizer. We apply weight decay only to the neural network parameters. For all models and datasets, we use the validation set to search over the β_{reg} regularization parameter $\beta_{\text{reg}} \in \{0.01, 0.2, 0.5, 1.0\}$ and the amount of weight decay $\in \{10^{-3}, 10^{-4}\}$. For the DGP/DSPP models we only consider 2-layer models with $W = 5$. The rest of the training details (learning rate, number of epochs, mini-batch sizes, etc.) match those outlined in Sec. A.2.

B TIME AND SPACE COMPLEXITY

We briefly describe the time and space complexity of 2-layer univariate DSPP models that utilize quadrature rule QR3. (Extending to deeper and multivariate models is straightforward.) Our analysis is similar to that of doubly-stochastic Deep Gaussian Processes (Salimbeni and Deisenroth, 2017). In particular, when using QR3, the running time and space complexities of DSPP are identical to that of the doubly stochastic DGP if the number of quadrature sites S for DSPP is taken to be equal to the number of samples used for the DGP.

We first note that computing the marginal distribution $q(f(\mathbf{x}))$ of a *single-layer* sparse Gaussian Process—as in Eqn. 10—is $\mathcal{O}(M^3)$. Both the predictive mean (Eqn. 7) and variance (Eqn. 8) require computing $\mathbf{K}_{MM}^{-1}$ which is a cubic operation. After this operation all other terms can be computed in $\mathcal{O}(M^2)$ time. If the inverse (or, more practically, its Cholesky factor) is cached, all subsequent predictive distributions also require $\mathcal{O}(M^2)$ time.

Training complexity. Each DSPP training iteration requires computing Eqn. 21. Again, let W be the width of the hidden GP layer, S be the number of quadrature points, and M be the number of inducing points (which, for simplicity, we assume is the same for both layers). The right term in Eqn. 21 is the sum of $W + 1$ KL divergences between M-dimensional multivariate Normal distributions (one for each hidden-layer GP and one for the final-layer GP), for a total of $\mathcal{O}(WM^3)$ computation. To compute the log likelihood term, we compute $q(g_1(\mathbf{x})), \ldots, q(g_W(\mathbf{x}))$, S quadrature points $\hat{\mathbf{g}}_1, \ldots, \hat{\mathbf{g}}_S$, and $q(f(\hat{\mathbf{g}}_1)), \ldots, q(f(\hat{\mathbf{g}}_S))$. This requires $\mathcal{O}(WM^3)$ computation for the $\mathbf{K}_{MM}^{-1}$ terms of each GP, and then multiplies against W matrices of size $M \times B$, where B is the minibatch size. In total the computational complexity is $\mathcal{O}(WM^3 + (S + B)WM^2)$. The space complexity is $\mathcal{O}(WM^2 + (S + W)MB)$—the size of storing each $\mathbf{K}_{MM}^{-1}$ as well as the additional vectors in Eqns. 7 and 8 for each marginal distribution $q(f(\hat{\mathbf{g}}_i))$.

Prediction complexity. We use nearly the same set of computations at test time as we do during training. The only difference is that, since the parameters are constant, we do not need to recompute the $\mathbf{K}_{MM}^{-1}$ matrices at every iteration. Consequentially, after the one-time $\mathcal{O}(M^3)$ cost to compute these inverses, the remaining terms can be computed in $\mathcal{O}((S + B)WM^2)$ time. The space complexity is still $\mathcal{O}(WM^2 + (S + W)MB)$.

C ADDITIONAL RESULTS

Quadrature weights. In Fig. 9 we depict a histogram of learned quadrature weights for 96 DSPPs trained using quadrature rule QR3. In particular we follow the experimental procedure described in Sec. A.2 with $S = 10$ quadrature points. We average over 3 train/test/validation splits for the 8 smallest UCI regression datasets with 4 different values of β_{reg}, for a total of 96 model runs. We then depict distributions over the largest, second largest, and third largest quadrature weights. We see that while a significant amount of the probability mass is put on the leading one or two mixture components, DSPP training does not result in degenerate weights: the final predictive distribution reflects a diversity of mean and variance functions.

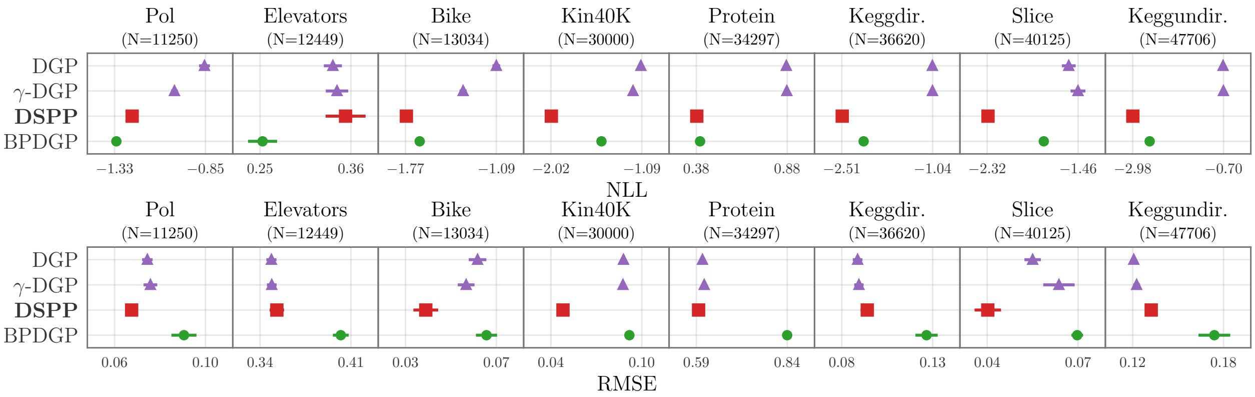

Figure 8: We depict negative log likelihoods (NLL, top) and root mean squared error (RMSE, bottom) for predictive deep GPs trained with biased Monte Carlo sampling (instead of learned quadrature weights). Results are averaged over five random train/test/validation splits.

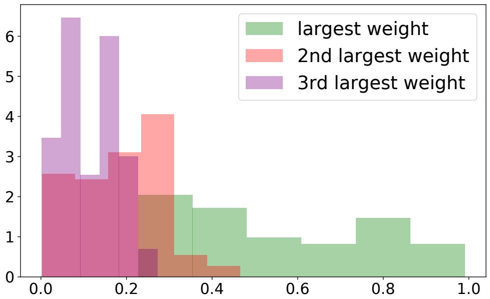

Figure 9: We depict histograms over leading quadrature weights $\omega^{(s)}$ for 96 DSPPs trained on a mixture of univariate regression datasets.

Using MC integration instead of quadrature. Rather than using a quadrature scheme or learned weights to evaluate the integral in Eq. 15, we could alternatively use Monte Carlo sampling. As described in Sec. 3, this would result in a biased estimate of the predictive objective function. In Fig. 8 we depict the NLL and RMSE of DSPP models that use biased MC sampling to evaluate Eq. 15 rather than quadrature. As this approach bypasses the finite mixture approximation made by DSPP models, we refer to this class of model as Biased Predictive Deep GPs (or **BPDGPs**). These models are trained following the procedure described in Sec. A.2.[8] We find that DSPP models (trained with QR3) tend to outperform the biased BPDGP models, both in terms of NLL and RMSE. In fact, the biased models achieve even worse RMSE than single-layer SVGP or PPGPR models.

Quadrature ablation study. Table 4 displays the average NLL, RMSE, and CRPS results across all datasets and train/test/validation splits.

Univariate regression. Table 5 summarizes of Figs. 2, 3 and 4. It displays the average NLL, RMSE, and CRPS results across datasets and train/test/validation splits.

Multivariate regression. Fig. 10 displays root mean squared error of the multivariate models on all datasets. Table 6 summarizes the results of Fig. 5 and Fig. 10.

Multilayer models. Fig. 11 displays the CRPS across 4 datasets (Kin40k, Protein, Keggdirected, and Slice) for models with multiple layers. Table 7 summarizes the results of Fig. 6 and Fig. 11.

Comparison with Deep Kernel Learning. Fig. 12 displays the CRPS across 4 datasets (Kin40k, Protein, Keggdirected, and Slice) for models augmented with neural networks (deep kernel learning). Table 8 summarizes the results of Fig. 7 and Fig. 12.

Results compilations. Tables 9, 10, and 11 report numbers for all experiments in Sec. 5. Numbers are averages ± standard errors over train/test/validation splits.

[8] Integrals are evaluated with 32 MC samples.

Table 4: Average NLL, RMSE, and CRPS of different quadrature rules (lower is better). Averages are aggregated across the smallest 8 UCI datasets and train/test/validation splits. See Sec. 5.1 for details.

	DSPP-QR1	DSPP-QR2	DSPP-QR3
NLL	-1.460	-1.444	$\mathbf{-1.509}$
RMSE	0.173	0.175	**0.171**
CRPS	0.077	0.079	**0.076**

Table 5: Average NLL, MRMSE, and RMSE for univariate models (lower is better). Averages are aggregated across the 12 univariate UCI datasets and train/test/validation splits. See Sec. 5.2 for details.

	OD-SVGP	PPGPR	DGP	γ-DGP	DSPP
NLL	-0.383	-0.730	-0.450	-0.485	$\mathbf{-1.198}$
RMSE	0.242	0.237	0.236	0.238	**0.231**
CRPS	0.130	0.117	0.124	0.122	**0.111**

Table 6: Average NLL, MRMSE, and RMSE for multivariate models (lower is better). Averages are aggregated across the five multivariate datasets and train/test/validation splits. See Sec. 5.3 for details.

	SVGP	PPGPR	γ-DGP	DGP	DSPP
NLL	-0.179	-0.889	-0.379	-0.240	$\mathbf{-1.058}$
MRMSE	0.198	0.255	0.213	0.188	**0.185**
RMSE	0.608	0.790	0.670	0.583	**0.576**

Table 7: Average NLL, RMSE, and CRPS for multi-layer models (lower is better). Averages are aggregated across 4 of the medium-sized UCI datasets (Kin40K, Protein, Keggdirected, Slice) and train/test/validation splits. See Sec. 5.4 for details.

	PPGPR (1L)	DGP (2L)	DSPP (2L)	DGP (3L)	DSPP (3L)
NLL	-0.889	-0.705	-1.612	-0.810	$\mathbf{-1.737}$
RMSE	0.203	0.210	0.195	0.214	**0.193**
CRPS	0.096	0.104	0.085	0.098	**0.084**

Table 8: Average NLL, RMSE, and CRPS of neural-network modulated (DKL) model variants. Averages are aggregated across the 4 UCI datasets (Kin40k, Protein, Keggdirected, and Slice) and train/test/validation splits. See Sec. 5.5 for details.

	SVGP	NN+SVGP	PPGPR	NN+PPGPR	DGP	NN+DGP	DSPP	NN+DSPP
NLL	-0.456	-0.897	-0.889	-1.231	-0.705	-0.948	$\mathbf{-1.608}$	-1.485
RMSE	0.219	0.184	0.203	0.198	0.210	**0.183**	0.194	0.194
CRPS	0.119	0.096	0.096	0.091	0.104	0.096	0.085	**0.081**

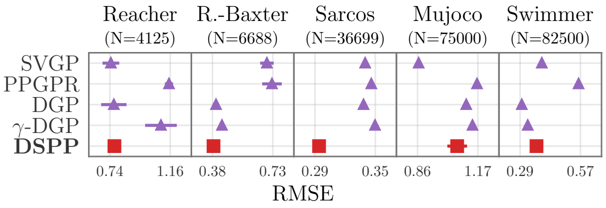

Figure 10: We depict root mean squared errors (RMSEs) for the 5 multivariate regression datasets in Sec. 5.3 (lower is better). Results are averaged over five random train/test/validation splits.

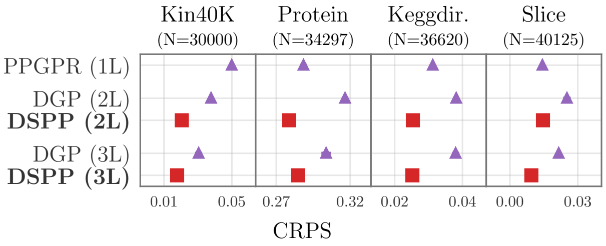

Figure 11: We depict the Continuous Ranked Probability Score (CRPS) for the multi-layer experiment in Sec. 5.4 (lower is better). Results are averaged over five random train/test/validation splits (for 3-layer models) and ten splits otherwise.

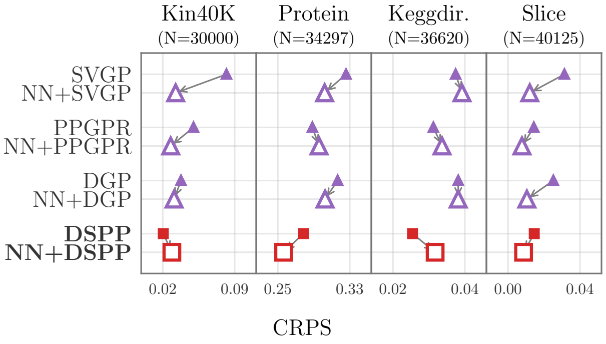

Figure 12: We depict the Continuous Ranked Probability Score (CRPS) of neural-network modulated (deep kernel learning) variants of GP/DGP/DSPP models (lower is better). Results are averaged over ten random train/test/validation splits. See Sec. 5.5 for details.

Table 9: A compilation of all UCI results from Secs. 5.1, 5.2, and 5.4. For each metric and dataset we bold the result for the best performing method (lower is better for all metrics). ± indicates standard error.

Metric	Dataset	OD-SVGP	PPGPR	γ-DGP (2L)	DGP (2L)	DGP (3L)	DSPP-QR1 (2L)	DSPP-QR2 (2L)	DSPP-QR3 (2L)	DSPP-QR3 (3L)
NLL	Pol	-0.723 ± 0.006	-1.090 ± 0.009	-1.014 ± 0.009	-0.855 ± 0.014	—	$\mathbf{-1.236 \pm 0.005}$	-1.223 ± 0.006	$\mathbf{-1.237 \pm 0.008}$	—
	Elevators	0.448 ± 0.010	0.368 ± 0.011	$\mathbf{0.343 \pm 0.007}$	$\mathbf{0.338 \pm 0.005}$	—	$\mathbf{0.346 \pm 0.011}$	$\mathbf{0.355 \pm 0.013}$	$\mathbf{0.354 \pm 0.012}$	—
	Bike	-0.824 ± 0.009	-1.426 ± 0.010	-1.339 ± 0.013	-1.090 ± 0.015	—	$\mathbf{-1.732 \pm 0.021}$	-1.717 ± 0.021	$\mathbf{-1.763 \pm 0.014}$	—
	Kin40K	-0.830 ± 0.004	-1.284 ± 0.005	-1.180 ± 0.006	-1.100 ± 0.004	-1.292 ± 0.024	-1.850 ± 0.034	-1.813 ± 0.039	-2.016 ± 0.012	$\mathbf{-2.133 \pm 0.011}$
	Protein	0.892 ± 0.006	0.743 ± 0.008	0.878 ± 0.005	0.875 ± 0.004	0.814 ± 0.005	$\mathbf{0.395 \pm 0.009}$	0.434 ± 0.012	$\mathbf{0.382 \pm 0.006}$	0.407 ± 0.014
	Keggdir.	-1.057 ± 0.018	-1.575 ± 0.015	-1.043 ± 0.018	-1.045 ± 0.016	-1.045 ± 0.031	-2.454 ± 0.009	-2.474 ± 0.011	-2.503 ± 0.016	$\mathbf{-2.556 \pm 0.029}$
	Slice	-1.673 ± 0.013	-1.438 ± 0.057	-1.461 ± 0.035	-1.549 ± 0.033	-1.719 ± 0.009	-2.194 ± 0.074	-2.146 ± 0.069	-2.312 ± 0.019	$\mathbf{-2.666 \pm 0.020}$
	Keggundir.	-0.712 ± 0.006	-1.801 ± 0.013	-0.703 ± 0.006	-0.712 ± 0.006	—	$\mathbf{-2.959 \pm 0.013}$	$\mathbf{-2.964 \pm 0.012}$	$\mathbf{-2.976 \pm 0.020}$	—
	3Droad	0.231 ± 0.014	-0.297 ± 0.003	0.216 ± 0.003	0.241 ± 0.003	—	—	—	$\mathbf{-0.488 \pm 0.006}$	—
	Song	1.168 ± 0.001	1.103 ± 0.001	1.164 ± 0.001	1.162 ± 0.001	—	—	—	$\mathbf{0.670 \pm 0.045}$	—
	Buzz	0.044 ± 0.002	-0.047 ± 0.001	0.002 ± 0.002	0.001 ± 0.002	—	—	—	$\mathbf{-0.209 \pm 0.014}$	—
	Houseelectric	-1.559 ± 0.002	-2.020 ± 0.003	-1.686 ± 0.003	-1.671 ± 0.003	—	—	—	$\mathbf{-2.281 \pm 0.003}$	—
RMSE	Pol	0.109 ± 0.001	0.077 ± 0.001	0.076 ± 0.002	0.074 ± 0.001	—	$\mathbf{0.065 \pm 0.001}$	$\mathbf{0.064 \pm 0.002}$	0.067 ± 0.001	—
	Elevators	0.370 ± 0.003	0.361 ± 0.003	$\mathbf{0.349 \pm 0.002}$	$\mathbf{0.349 \pm 0.002}$	—	$\mathbf{0.350 \pm 0.002}$	$\mathbf{0.351 \pm 0.003}$	$\mathbf{0.353 \pm 0.003}$	—
	Bike	0.097 ± 0.002	0.060 ± 0.001	0.057 ± 0.002	0.062 ± 0.002	—	$\mathbf{0.043 \pm 0.004}$	$\mathbf{0.044 \pm 0.004}$	$\mathbf{0.039 \pm 0.003}$	—
	Kin40K	0.109 ± 0.001	0.126 ± 0.001	0.088 ± 0.001	0.088 ± 0.000	0.075 ± 0.003	0.057 ± 0.002	0.060 ± 0.003	0.048 ± 0.001	$\mathbf{0.042 \pm 0.001}$
	Protein	0.591 ± 0.003	$\mathbf{0.569 \pm 0.002}$	0.612 ± 0.002	0.607 ± 0.002	0.643 ± 0.005	0.599 ± 0.003	0.609 ± 0.002	0.596 ± 0.003	0.606 ± 0.003
	Keggdir.	$\mathbf{0.085 \pm 0.001}$	$\mathbf{0.087 \pm 0.001}$	0.089 ± 0.001	0.089 ± 0.001	$\mathbf{0.089 \pm 0.003}$	0.092 ± 0.002	0.094 ± 0.002	0.094 ± 0.002	0.094 ± 0.005
	Slice	0.043 ± 0.001	$\mathbf{0.032 \pm 0.001}$	0.064 ± 0.003	0.055 ± 0.001	0.048 ± 0.001	0.049 ± 0.007	0.047 ± 0.008	0.040 ± 0.002	$\mathbf{0.030 \pm 0.004}$
	Keggundir.	$\mathbf{0.119 \pm 0.001}$	0.123 ± 0.001	0.123 ± 0.001	0.121 ± 0.001	—	0.132 ± 0.001	0.132 ± 0.001	0.132 ± 0.001	—
	3Droad	0.303 ± 0.004	0.304 ± 0.001	0.322 ± 0.001	0.322 ± 0.001	—	—	—	$\mathbf{0.296 \pm 0.002}$	—
	Song	0.778 ± 0.001	$\mathbf{0.770 \pm 0.001}$	0.782 ± 0.001	0.780 ± 0.001	—	—	—	0.820 ± 0.008	—
	Buzz	0.256 ± 0.001	0.283 ± 0.001	$\mathbf{0.244 \pm 0.001}$	$\mathbf{0.244 \pm 0.000}$	—	—	—	0.247 ± 0.001	—
	Houseelectric	0.050 ± 0.000	0.046 ± 0.000	0.046 ± 0.000	0.046 ± 0.000	—	—	—	$\mathbf{0.042 \pm 0.000}$	—
CRPS	Pol	0.059 ± 0.000	0.040 ± 0.000	0.040 ± 0.000	0.045 ± 0.001	—	$\mathbf{0.033 \pm 0.000}$	0.034 ± 0.000	$\mathbf{0.034 \pm 0.000}$	—
	Elevators	0.203 ± 0.001	0.195 ± 0.002	$\mathbf{0.186 \pm 0.001}$	$\mathbf{0.186 \pm 0.001}$	—	0.189 ± 0.001	0.190 ± 0.001	0.191 ± 0.001	—
	Bike	0.051 ± 0.001	0.028 ± 0.000	0.027 ± 0.000	0.034 ± 0.001	—	$\mathbf{0.021 \pm 0.001}$	0.021 ± 0.001	$\mathbf{0.019 \pm 0.001}$	—
	Kin40K	0.056 ± 0.000	0.050 ± 0.000	0.036 ± 0.000	0.038 ± 0.000	0.030 ± 0.001	0.024 ± 0.001	0.025 ± 0.001	0.020 ± 0.000	$\mathbf{0.018 \pm 0.000}$
	Protein	0.317 ± 0.001	0.288 ± 0.001	0.315 ± 0.001	0.317 ± 0.001	0.304 ± 0.002	$\mathbf{0.281 \pm 0.002}$	0.288 ± 0.001	$\mathbf{0.279 \pm 0.002}$	0.285 ± 0.002
	Keggdir.	0.037 ± 0.000	0.031 ± 0.000	0.037 ± 0.000	0.038 ± 0.000	0.038 ± 0.001	$\mathbf{0.025 \pm 0.000}$	$\mathbf{0.026 \pm 0.000}$	$\mathbf{0.025 \pm 0.000}$	$\mathbf{0.025 \pm 0.001}$
	Slice	0.021 ± 0.000	0.014 ± 0.000	0.026 ± 0.000	0.025 ± 0.001	0.022 ± 0.000	0.019 ± 0.003	0.019 ± 0.003	0.015 ± 0.000	$\mathbf{0.009 \pm 0.000}$
	Keggundir.	0.051 ± 0.000	0.036 ± 0.000	0.050 ± 0.000	0.051 ± 0.000	—	$\mathbf{0.027 \pm 0.000}$	$\mathbf{0.027 \pm 0.000}$	$\mathbf{0.027 \pm 0.000}$	—
	3Droad	0.163 ± 0.002	0.138 ± 0.000	0.161 ± 0.000	0.164 ± 0.001	—	—	—	$\mathbf{0.128 \pm 0.001}$	—
	Song	0.434 ± 0.000	$\mathbf{0.422 \pm 0.000}$	0.433 ± 0.000	0.432 ± 0.000	—	—	—	0.448 ± 0.005	—
	Buzz	0.135 ± 0.000	0.134 ± 0.000	0.129 ± 0.000	0.130 ± 0.000	—	—	—	$\mathbf{0.126 \pm 0.000}$	—
	Houseelectric	0.026 ± 0.000	0.022 ± 0.000	0.023 ± 0.000	0.023 ± 0.000	—	—	—	$\mathbf{0.018 \pm 0.000}$	—

Table 10: A compilation of all multivariate results from Sec. 5.3. For each metric and dataset we bold the result for the best performing method (lower is better for all metrics). ± indicates standard error.

Metric	Dataset	SVGP	PPGPR	γ-DGP (2L)	DGP (2L)	DSPP-QR3 (2L)
NLL	Reacher	0.460 ± 0.048	-0.454 ± 0.005	0.215 ± 0.032	0.484 ± 0.032	$\mathbf{-0.491 \pm 0.004}$
	R.-Baxter	0.112 ± 0.042	-0.498 ± 0.010	-0.728 ± 0.012	-0.303 ± 0.036	$\mathbf{-0.910 \pm 0.005}$
	Sarcos	-0.703 ± 0.001	-1.032 ± 0.002	-0.700 ± 0.002	-0.694 ± 0.003	$\mathbf{-1.169 \pm 0.003}$
	Mujoco	0.141 ± 0.002	-0.563 ± 0.001	0.335 ± 0.007	0.339 ± 0.005	$\mathbf{-0.636 \pm 0.010}$
	Swimmer	-0.905 ± 0.011	-1.898 ± 0.002	-1.018 ± 0.003	-1.025 ± 0.007	$\mathbf{-2.085 \pm 0.002}$
MRMSE	Reacher	$\mathbf{0.217 \pm 0.009}$	0.330 ± 0.005	0.323 ± 0.017	$\mathbf{0.237 \pm 0.015}$	$\mathbf{0.230 \pm 0.007}$
	R.-Baxter	0.247 ± 0.007	0.263 ± 0.010	0.140 ± 0.001	$\mathbf{0.126 \pm 0.003}$	$\mathbf{0.121 \pm 0.003}$
	Sarcos	0.124 ± 0.000	0.126 ± 0.000	0.128 ± 0.000	0.123 ± 0.000	$\mathbf{0.108 \pm 0.001}$
	Mujoco	$\mathbf{0.286 \pm 0.001}$	0.383 ± 0.001	0.373 ± 0.002	0.362 ± 0.002	0.352 ± 0.008
	Swimmer	0.115 ± 0.001	0.172 ± 0.000	0.100 ± 0.001	$\mathbf{0.092 \pm 0.001}$	0.114 ± 0.004
RMSE	Reacher	$\mathbf{0.747 \pm 0.029}$	1.151 ± 0.014	1.095 ± 0.055	$\mathbf{0.767 \pm 0.044}$	$\mathbf{0.771 \pm 0.015}$
	R.-Baxter	0.696 ± 0.020	0.725 ± 0.028	0.435 ± 0.005	$\mathbf{0.401 \pm 0.009}$	$\mathbf{0.386 \pm 0.010}$
	Sarcos	0.340 ± 0.001	0.346 ± 0.001	0.350 ± 0.001	0.338 ± 0.001	$\mathbf{0.294 \pm 0.001}$
	Mujoco	$\mathbf{0.866 \pm 0.002}$	1.166 ± 0.004	1.142 ± 0.008	1.110 ± 0.006	1.064 ± 0.025
	Swimmer	0.391 ± 0.002	0.561 ± 0.001	0.326 ± 0.001	$\mathbf{0.298 \pm 0.003}$	0.366 ± 0.012

Table 11: A compilation of all deep kernel learning results (UCI datasets) from Sec. 5.5. For each metric and dataset we bold the result for the best performing method (lower is better for all metrics). ± indicates standard error.

Metric	Dataset	SVGP	NN+SVGP	PPGPR	NN+PPGPR	DGP (2L)	NN+DGP	DSPP	NN+DSPP
NLL	Kin40K	-0.414 ± 0.002	-1.235 ± 0.004	-1.284 ± 0.005	-1.483 ± 0.002	-1.100 ± 0.004	-1.273 ± 0.005	$\mathbf{-2.016 \pm 0.012}$	-1.458 ± 0.011
	Protein	0.902 ± 0.003	0.885 ± 0.005	0.743 ± 0.008	0.879 ± 0.017	0.875 ± 0.004	0.890 ± 0.004	0.394 ± 0.008	$\mathbf{0.278 \pm 0.010}$
	Keggdir.	-1.045 ± 0.017	-1.035 ± 0.012	-1.575 ± 0.015	-1.505 ± 0.015	-1.045 ± 0.016	-1.044 ± 0.014	$\mathbf{-2.498 \pm 0.015}$	-2.034 ± 0.022
	Slice	-1.267 ± 0.003	-2.204 ± 0.022	-1.438 ± 0.057	$\mathbf{-2.815 \pm 0.009}$	-1.549 ± 0.033	-2.363 ± 0.016	-2.312 ± 0.019	-2.728 ± 0.010
RMSE	Kin40K	0.147 ± 0.001	0.052 ± 0.001	0.126 ± 0.001	0.062 ± 0.002	0.088 ± 0.000	$\mathbf{0.048 \pm 0.001}$	$\mathbf{0.048 \pm 0.001}$	0.064 ± 0.004
	Protein	0.594 ± 0.002	0.578 ± 0.003	0.569 ± 0.002	0.599 ± 0.003	0.607 ± 0.002	0.580 ± 0.002	0.595 ± 0.004	$\mathbf{0.559 \pm 0.004}$
	Keggdir.	$\mathbf{0.086 \pm 0.001}$	$\mathbf{0.086 \pm 0.002}$	$\mathbf{0.087 \pm 0.001}$	0.107 ± 0.004	0.089 ± 0.001	$\mathbf{0.085 \pm 0.001}$	0.095 ± 0.002	0.116 ± 0.008
	Slice	0.051 ± 0.001	$\mathbf{0.019 \pm 0.001}$	0.032 ± 0.001	0.023 ± 0.003	0.055 ± 0.001	$\mathbf{0.018 \pm 0.000}$	0.040 ± 0.002	0.038 ± 0.007
CRPS	Kin40K	0.082 ± 0.000	0.033 ± 0.000	0.050 ± 0.000	0.028 ± 0.000	0.038 ± 0.000	0.031 ± 0.000	$\mathbf{0.020 \pm 0.000}$	0.029 ± 0.001
	Protein	0.326 ± 0.001	0.302 ± 0.002	0.288 ± 0.001	0.296 ± 0.001	0.317 ± 0.001	0.303 ± 0.002	0.278 ± 0.002	$\mathbf{0.256 \pm 0.003}$
	Keggdir.	0.037 ± 0.000	0.039 ± 0.000	0.031 ± 0.000	0.034 ± 0.000	0.038 ± 0.000	0.038 ± 0.000	$\mathbf{0.025 \pm 0.000}$	0.032 ± 0.001
	Slice	0.031 ± 0.000	0.012 ± 0.000	0.014 ± 0.000	$\mathbf{0.008 \pm 0.000}$	0.025 ± 0.001	0.010 ± 0.000	0.015 ± 0.000	0.009 ± 0.000