Guided Deep Kernel Learning

Abstract

Combining Gaussian processes with the expressive power of deep neural networks is commonly done nowadays through deep kernel learning (DKL). Unfortunately, due to the kernel optimization process, this often results in losing their Bayesian benefits. In this study, we present a novel approach for learning deep kernels by utilizing infinite-width neural networks. We propose to use the Neural Network Gaussian Process (NNGP) model as a guide to the DKL model in the optimization process. Our approach harnesses the reliable uncertainty estimation of the NNGPs to adapt the DKL target confidence when it encounters novel data points. As a result, we get the best of both worlds, we leverage the Bayesian behavior of the NNGP, namely its robustness to overfitting, and accurate uncertainty estimation, while maintaining the generalization abilities, scalability, and flexibility of deep kernels. Empirically, we show on multiple benchmark datasets of varying sizes and dimensionality, that our method is robust to overfitting, has good predictive performance, and provides reliable uncertainty estimations.

1 INTRODUCTION

Gaussian processes (GPs) are an effective Bayesian non-parametric family of models. They have several appealing features, such as tractable inference, accurate uncertainty estimation, and the ability to generalize well from small datasets [Rasmussen and Williams, 2006, Snell and Zemel, 2021, Achituve et al., 2021b]. In GPs, the kernel function is the crucial factor that determines their performance, as it measures the similarity between data points and significantly impacts which functions the model considers probable. Standard kernels, such as RBF kernels, perform well on certain learning problems, but they are inadequate for complex data modalities, like images and texts, failing to capture the desired semantic similarity. One appealing solution is to combine GPs with the expressive power of Neural Networks (NNs). There are two popular ways to achieve that. The first is through learning deep kernels, and the second is through kernels that correspond to infinite-width networks. In what follows we present both approaches, their limitations, and our proposed approach that combines the two.

One popular way to combine GPs and NNs is through *deep kernel learning* (DKL) [Calandra et al., 2016, Wilson et al., 2016a]. DKL uses a standard kernel over an embedding learned by a neural network, combining the tractable inference of GPs with the expressive power of deep neural networks (DNNs). Unfortunately, despite appearing to be a natural way to combine the benefits of GPs and DNNs, DKL often falls short of expectations in practice. A recent study found that deep kernels can severely overfit, sometimes even worse than standard NNs [Ober et al., 2021]. This work suggests that the DKL overfitting is caused by the optimization process "over-correlating" the data points.

An alternative way to link DNNs and GPs, without relying on DKL, is through the equivalence between GPs and *infinite-width* deep neural networks [Neal, 1996, Lee et al., 2018, de G. Matthews et al., 2018, Garriga-Alonso et al., 2019, Novak et al., 2019, Yang, 2019]. Specifically, consider the distribution over DNN weights when they are initialized i.i.d. As the width of the DNN layers increases to infinity, the distribution of functions represented by the NN converges to a Gaussian process. Importantly, that GP has a kernel function that can be computed efficiently despite having an infinite width. The main advantage of this approach is clear - it allows us to apply tractable Bayesian inference to highly expressive neural networks of infinite width. And, as the structure of DNNs provide valuable inductive biases for many data modalities, they can generate a corresponding kernel that is better suited to various data modalities.

This approach, however, also has several drawbacks that

Submitted to the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023). **To be used for reviewing only**.

hinder its widespread adoption. First and foremost, in many cases, these models underperform standard NNs that were optimized for a specific task [Novak et al., 2019]. One possible explanation for this is that the success of DNNs is connected to the implicit bias in the optimization process (e.g. [Vardi and Shamir, 2021]), which can not be captured by them. Second, the evaluation of the kernel in training and inference time can be costly. This is partially due to the fact that the NN kernel needs to be computed for every pair of data points, in comparison to DKL where we run the network on each datum once before applying a standard kernel. Finally, it is challenging to incorporate established mechanisms such as inducing point techniques with these types of models. This is in contrast to DKLs which are more flexible and easier to scale.

The limitations of current solutions raise the question: *How can we combine GPs with NNs without compromising performance or uncertainty estimation?*

This paper proposes a solution to the above question, which we call *Guided Deep Kernel Learning* (GDKL). GDKL combines the benefits of DKL with NNGPs, by leveraging the uncertainty estimation of NNGPs to guide the DKL optimization process. To this end, we propose a novel procedure to optimize deep kernels by having them match the distribution of the NNGP's latent function given the target value. For example, consider a regression task, the DKL will try to match a Gaussian centered near the target with an adaptive level of certainty that depends on the NNGP. We show that this approach achieves the best of both worlds. It enjoys the flexibility, scalability, and predictive capabilities of DKL, while retaining the Bayesian benefits of GPs. Namely, our method can estimate uncertainty more reliably and is drastically less prone to overfitting than DKL, without sacrificing performance. The experiments show the superiority of our method against natural baseline methods on several benchmark datasets in terms of both performance and uncertainty quantification.

This paper makes the following novel contributions. (i) We propose GDKL, a novel method to train deep kernels having their uncertainty calibrated by infinite-width networks; (ii) GDKL allows to perform either exact inference or approximate inference using common inducing point techniques; (iii) we demonstrate the benefits of GDKL over baseline methods for small- to mid-sized benchmark datasets with low and high data dimensionality. We conclude that GDKL generalizes well, can estimate uncertainty more reliably, and is significantly more robust against overfitting compared to standard deep kernels and competing methods.

2 BACKGROUND

Notations. We denote scalars with lower-case letters (e.g., x), vectors with bold lower-case letters, (e.g., $\mathbf{x}$), and matrices with bold capital letters (e.g., $\mathbf{X}$). Given a dataset $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_n, \mathbf{y}_n)\}$, we denote by $\mathbf{X} \in \mathbb{R}^{n \times d}$ and $\mathbf{Y} \in \mathbb{R}^{n \times c}$ the design and label matrices whose i^{th} row is $\mathbf{x}_i$ and $\mathbf{y}_i$ respectively.

Gaussian Processes. Gaussian processes are a family of Bayesian non-parametric models. GPs assume that the mapping from input points to the target values is via latent functions $\mathcal{F} = \{f^1, ..., f^c\}$. In this study, we assume independence between the latent function values. Consider a single output dimension process $f(\cdot)$, a GP is fully specified by the mean function $m(\mathbf{x})$ and the covariance function $k(\mathbf{x}, \mathbf{x}')$. We denote it by $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}),\ k(\mathbf{x}, \mathbf{x}'))$. The mean $m(\mathbf{x})$ is commonly taken to be the constant zero function, and the kernel $k(\mathbf{x}, \mathbf{x}')$ is a positive semi-definite function. The kernel defines the correlation between function values at different input locations. Thus, it is the main contributing factor in predicting on novel inputs. One of the major benefits of GPs is that in regression tasks with Gaussian noise, $p(y_i|f(\mathbf{x}_i)) = \mathcal{N}(f(\mathbf{x}_i), \sigma_n^2)$, the inference has a closed-form Gaussian solution. Specifically, we have analytical expressions for the posterior $p(f_*|\mathbf{x}_*, \mathcal{D})$ and the marginal $p(y_*|\mathbf{x}_*, \mathcal{D})$ where $\mathcal{D}$ is the training data and $\mathbf{x}_*$ is a test data point. The hyper-parameters of the GPs, which we will refer to as parameters in this study, are commonly optimized using the marginal likelihood. Here, we promote the use of the predictive distribution to learn them. Several studies considered this approach in the literature (e.g., [Jankowiak et al., 2020, Snell and Zemel, 2021, Achituve et al., 2021b, Lotfi et al., 2022]). Usually, this objective leads to better predictive abilities, yet as we will show, it is not robust against overfitting when training deep kernels.

Deep Kernel Learning In [Wilson et al., 2016a], the authors proposed to combine deep neural networks with GPs by applying a GP on the representation learned by a NN. For example, consider the RBF kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-||\mathbf{x} - \mathbf{x}'||^2/2\ell^2)$ (although any other kernel can be used), they proposed the following kernel $k_\theta(\mathbf{x}, \mathbf{x}') = \exp(-||g_\theta(\mathbf{x}) - g_\theta(\mathbf{x}')||^2/2\ell^2)$ where g_θ is a NN with parameters θ. They then trained θ to maximize the log marginal likelihood $\log(p(\mathbf{y}|\mathbf{X}))$ using the closed-form expression for regression problems. Later works extended this approach to classification [Linderman et al., 2015, Wilson et al., 2016b, Milios et al., 2018, Achituve et al., 2021a].

Infinite width networks. Studying the behavior of NNs in the infinite-width limit has its roots in the seminal work of Neal [1996]. It was shown that at initialization (with proper tuning), the distribution over functions represented by a single hidden layer NN converges to a GP as the width increases to infinity. This approach was later extended to infinite deep NNs as well [Lee et al., 2018, de G. Matthews et al., 2018]. This means that at the infinite-width limit, the Bayesian neural network inference problem is reduced to GP

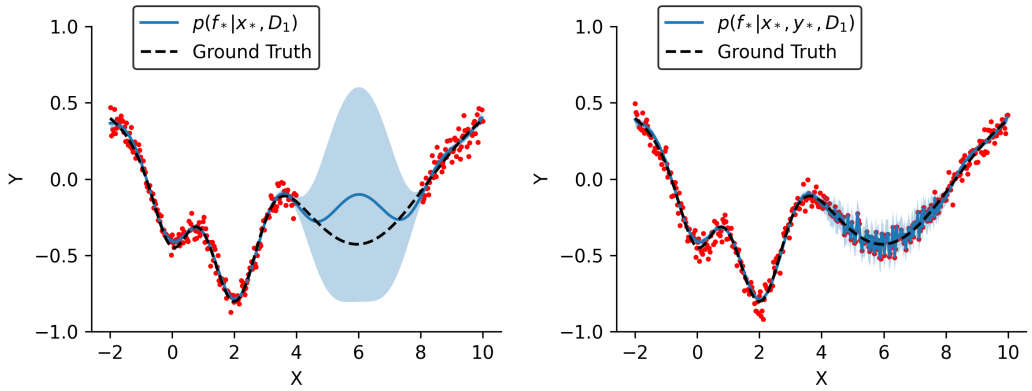

Figure 1: Illustrative example: (Left) Points in $\mathcal{D}_1$, the target function, and the GP prediction. Data contains a gap in [4,8] to demonstrate a low-confidence region. (Right) Points in $\mathcal{D}_2$ and the Gaussian objective $p(f_*|x_*, y_*, \mathcal{D}_1)$ for each point.

inference with a kernel defined by the neural network limit. In this study, we will refer to instances of this approach as the Neural Network Gaussian Process (NNGP). The kernel for a fully-connected network can be computed using the following recursive formula:

$$k^{(1)}(\mathbf{x}, \mathbf{x}') = \sigma_b^2 + \sigma_w^2 \cdot \frac{\mathbf{x}^T \mathbf{x}'}{d}$$
$$k^{(l+1)}(\mathbf{x}, \mathbf{x}') = \sigma_b^2 + \sigma_w^2 \mathbb{E}_{f \sim \mathcal{N}(0, \mathbf{K}^{(l)})}[\phi(f(\mathbf{x}))\phi(f(\mathbf{x}'))], \tag{1}$$

where σ_b^2, σ_w^2 are the variances of the biases and weights respectively, d and l are the input dimension and layer index respectively, and $\phi(\cdot)$ is the layer point-wise non-linear function. For some non-linear activations, such as sigmoidal, Gaussian, and Relu, the formula can be computed analytically [Williams, 1996, Cho and Saul, 2009]. In other cases, it can be approximated efficiently using Monte-Carlo methods as the expectation is over a two-dimensional Gaussian random variable [Novak et al., 2020]. Similarly, a kernel can be derived for other NN architectures, such as CNNs and RNNs [Garriga-Alonso et al., 2019, Novak et al., 2019, Yang, 2019].

Inducing points. One prominent limitation of GPs is the difficulty of doing exact inference on large datasets. Assuming we have a dataset with n points, exact inference requires storing and inverting an $n \times n$ matrix. This operation imposes a memory and run-time complexity of $\Omega(n^2)$. A commonly used method to improve the scalability of GPs is through the use of inducing points (e.g., [Titsias, 2009]). Inducing point methods define a set of pseudo-observations of size $m \ll n$, termed inducing locations. These locations may or may not be learned as part of the optimization process. Importantly, this mechanism allows us to control the size of the matrix to invert since in order to make predictions we can make all the costly operations only on these points instead of the actual dataset.

3 METHOD

We now present and explain our method. We will first describe our approach in the setting of exact GP inference (i.e., without inducing points approximation), then we will show how our framework can be easily generalized to include inducing points, i.e the sparse GP case. Incorporating inducing points into our framework allows our approach to handle a wide range of problems, from limited-sized datasets, where the overfitting of DKLs is most severe, to large-scale problems where the NNGPs are too computationally demanding to run. It is important to stress that we do not place any constraints on the NN architecture, unlike other existing solutions [Liu et al., 2021, Mallick et al., 2021, Ober et al., 2021, van Amersfoort et al., 2021]. For the sake of clarity, we will describe our method for the case of a single output. The generalization to the multi-outputs case is immediate and will be discussed afterward.

3.1 GUIDED DEEP KERNEL LEARNING

Assume we are given a dataset $\mathcal{D}$. We split it into a training set $\mathcal{D}_1$ and a validation set $\mathcal{D}_2$. Denote by p the NNGP model defined by an infinite-width neural network, and by q_θ the DKL model with parameters θ. Given a point $\mathbf{x}_*$ we denote by f_* the value of the latent GP function on $\mathbf{x}_*$, and denote by D_{KL} the Kullback–Leibler (KL) divergence.

To motivate our proposed approach, we first describe two possible objectives, a Bayesian distillation objective, and a predictive objective. Then, we present our final objective, which can be viewed as a combination of the two.

In distillation, we wish to train q_θ to mimic p. A natural way to achieve this goal is with the following objective:

$$\ell_{dist}(\theta) = \mathbb{E}_{\mathbf{x}_* \sim \mathcal{D}_2} D_{KL}[q_\theta(f_*|\mathbf{x}_*, \mathcal{D}_1) || p(f_*|\mathbf{x}_*, \mathcal{D}_1)]. \tag{2}$$

The objective in Eq. 2 tries to match the latent distribution

of the two models on an unseen data point from $\mathcal{D}_2$. Training q_θ may produce a model that behaves like the Bayesian NNGP model, but unfortunately, it will also inherit the sub-par predictive performance of the NNPG model.

Alternatively, we can try to optimize the predictive distribution of the DKL model:

$$\ell_{pred}(\theta) = \mathbb{E}_{\mathbf{x}_*, y_* \sim \mathcal{D}_2}[-\log\ q_\theta(y_*|\mathbf{x}_*, \mathcal{D}_1)]. \quad (3)$$

This objective tends to produce accurate predictions as it is directly optimized to predict y_*. However, it will not behave like a Bayesian model. Namely, it will suffer from the same overfitting issues as the standard DKL training does [Lotfi et al., 2022] which will result in an overestimated confidence. Appendix C.1 empirically demonstrates this claim and the previous one. We evaluated both loss terms on the UCI datasets Boston, Concrete, and Energy, and found that indeed these losses behave as we anticipated.

A possible middle ground between these two approaches is to optimize $q_\theta(y_*|\mathcal{D}_1, \mathbf{x}_*)$ to match the distribution of $p(f_*|\mathcal{D}_1, \mathbf{x}_*, y_*)$. The key difference is that in this case the latent variable f_* is also conditioned on the sample y_*:

$$\mathbb{E}_{\mathbf{x}_*, y_* \sim \mathcal{D}_2} D_{KL}[q_\theta(f_*|\mathbf{x}_*, \mathcal{D}_1)||p(f_*|\mathbf{x}_*, y_*, \mathcal{D}_1)]. \quad (4)$$

As the target latent distribution is conditioned on y_*, it will, for regression, take the form of a Gaussian centered near it, and the variance will be dependent on how confident $p(f_*|\mathbf{x}_*, y_*, \mathcal{D}_1)$ is. In Figure 1 we illustrate the usefulness of this objective on a toy problem. On the left panel we show the points in $\mathcal{D}_1$, the ground truth function, as well as the posterior $p(f_*|x_*, \mathcal{D}_1)$. We intentionally omitted points in the $[4, 8]$ domain from $\mathcal{D}_1$ to highlight areas where the GP is not confident. While the GP prediction in that area is not accurate, it appropriately assigns high uncertainty to its prediction. On the right panel, we show the points in $\mathcal{D}_2$ and for each point, we show the target objective $p(f_*|x_*, y_*, \mathcal{D}_1)$. We highlight two desired properties of our objective seen from this plot: When the GP is confident, the GP prediction is tight around the ground truth, and not centered around the noisy y_* samples. However, when the GP is not confident $p(f_*|x_*, y_*, \mathcal{D}_1)$ is centered around the noisy y_* samples with a much larger variance.

To gain further insight into our approach, consider estimating $\log\ p(y_*|\mathbf{x}_*, \mathcal{D}_1)$ using variational inference. While it has an analytical solution for a Gaussian likelihood, we can set it aside and derive the following evidence lower bound (ELBO):

$$\begin{aligned}
&\mathbb{E}_{\mathbf{x}_*, y_* \sim \mathcal{D}_2} \log\ p(y_*|\mathbf{x}_*, \mathcal{D}_1) = \\
&\mathbb{E}_{\mathbf{x}_*, y_* \sim \mathcal{D}_2} \log \int \frac{q_\theta(f_*|\mathcal{D}_1)}{q_\theta(f_*|\mathcal{D}_1)} p(y_*|f_*) p(f_*|\mathbf{x}_*, \mathcal{D}_1) df_* \geq \\
&\mathbb{E}_{\mathbf{x}_*, y_* \sim \mathcal{D}_2} \int q_\theta(f_*|\mathcal{D}_1) \log \frac{p(y_*|f_*) p(f_*|\mathbf{x}_*, \mathcal{D}_1)}{q_\theta(f_*|\mathcal{D}_1)} df_*.
\end{aligned} \quad (5)$$

Algorithm 1 Guided Deep Kernel Learning (*GDKL*)

Input: $\mathcal{D} = (\mathbf{X}, \mathbf{y})$ - the dataset; $\mathbf{K}$ - a pre-computed kernel of the NNGP on $\mathcal{D}$; T - number of training iterations; β - a hyper-parameter that scales the KL-divergence term.
Init θ, the parameters of the DKL.
For $i = 1, ..., T$:
- Randomly split $\mathcal{D}$ to $\mathcal{D}_1$ and $\mathcal{D}_2$, s.t. $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$ and $\mathcal{D}_1 \cap \mathcal{D}_2 = \{\emptyset\}$
- Construct $\mathbf{K}_{\mathcal{D}_1}$ from $\mathbf{K}$ by selecting the entries of examples from $\mathcal{D}_1$

For all $j \in \mathcal{D}_2$:
Exact expressions in the Appendix.
- Obtain the predictive posteriors $p(\mathbf{f}_j|\mathbf{x}_j, \mathcal{D}_1)$, and $q_\theta(\mathbf{f}_j|\mathbf{x}_j, \mathcal{D}_1)$
- $\mathcal{L}_j^{D_{KL}} \leftarrow D_{KL}[q_\theta(f_j|\mathbf{x}_j, \mathcal{D}_1)||p(f_j|\mathbf{x}_j, \mathcal{D}_1)]$
- $\mathcal{L}_j^{ELL} \leftarrow \mathbb{E}_{q_\theta(f_j|\mathbf{x}_j, \mathcal{D}_1)}[-\log\ p(y_j|f_j)]$

End for
- $\mathcal{L} \leftarrow \frac{1}{|\mathcal{D}_2|} \Sigma_{j=1}^{|\mathcal{D}_2|} \mathcal{L}_j^{ELL} + \beta \cdot \mathcal{L}_j^{D_{KL}}$.
- Compute $\nabla_\theta \mathcal{L}$ and perform update step.

End for

It is not hard to see that our objective is equivalent to maximizing the ELBO. Namely, q_θ is essentially trained as a variational distribution, similar to an encoder network in variational auto-encoders (VAEs) [Kingma and Welling, 2014]. It thus tries to "encode" the label, but does not directly predict it.

In Appendix B we show that the objective in Eq. 4 is also equivalent to the following:

$$\begin{aligned}
&\mathbb{E}_{\mathbf{x}_*, y_* \sim \mathcal{D}_2} \mathbb{E}_{q_\theta(f_*|\mathbf{x}_*, \mathcal{D}_1)}[-\log\ p(y_*|f_*)] \\
&+ D_{KL}[q_\theta(f_*|\mathbf{x}_*, \mathcal{D}_1)||p(f_*|\mathbf{x}_*, \mathcal{D}_1)].
\end{aligned} \quad (6)$$

This representation makes it clear that our objective is in fact a combination of Eq. 2 and Eq. 3. We can think of it as having a data term, which is comparable to the DKL marginal likelihood objective, plus a regularizer that prevents the model from over-fitting the training points.

Lastly, we introduce two modifications to our objective. We add a hyperparameter β that multiplies the D_{KL} term in Eq. 6. This allows us to control the balance between predictive training ($\beta = 0$) and distillation ($\beta \to \infty$). Second, we utilize the fact that GPs are non-parametric and perform a random split of $\mathcal{D}$ to different $\mathcal{D}_1$ and $\mathcal{D}_2$ sets at each iteration. We found that this approach led to better generalization compared to fixing these datasets. Hence the final objective is the following:

$$\begin{aligned}
L(\theta) =& \mathbb{E}_{\mathcal{D}_1, \mathcal{D}_2} \mathbb{E}_{\mathbf{x}_*, y_* \sim \mathcal{D}_2} \{\mathbb{E}_{q_\theta(f_*|\mathbf{x}_*, \mathcal{D}_1)}[-\log\ p(y_*|f_*)] \\
&+ \beta \cdot D_{KL}[q_\theta(f_*|\mathbf{x}_*, \mathcal{D}_1)||p(f_*|\mathbf{x}_*, \mathcal{D}_1)]\}.
\end{aligned} \quad (7)$$

Note that for the regression case, the D_{KL} term and the expected log-likelihood term have a closed-form solution. In

classification tasks, we can use approximations that involve a Gaussian likelihood, such as treating the classification problem as a regression problem, using transformed Dirichlet variables [Milios et al., 2018], or using the Pólya-Gamma augmentation [Polson et al., 2013, Achituve et al., 2021a]. In this study, we used the transformed Dirichlet variables technique. GDKL training procedure is illustrated in Algorithm 1.

Finally, we would like to note a few technical details. First, we can evaluate the kernel of the NNGP once and extract at each iteration only the relevant sub-matrices. Second, the extension to multi-output GPs results in additional summation over each output dimension. Third, our network and the NNGP network do not have to share the same architecture (besides the obvious difference in width) and we have complete freedom in choosing the architecture. Finally, to make predictions on novel data points we use the full dataset $\mathcal{D}$ using the standard GP formulas. Namely, when making predictions our model is as fast as standard DKL models.

3.2 GUIDED DEEP KERNEL LEARNING WITH INDUCING POINTS

Although the overfitting problem of DKLs is more acute in cases with limited data, it can still return overconfident predictions on large datasets. As such, we wish to extend our approach to larger datasets by incorporating inducing points. Denote by $\mathbf{Z}$ the set of m inducing locations, and by $\mathbf{u} = \mathbf{f}(\mathbf{Z})$ the function evaluation at these locations (i.e., the inducing variables). We follow the common practice [Hensman et al., 2013], and define the posterior now as $q_\theta(\mathbf{f}) = \int p_\theta(\mathbf{f}|\mathbf{u})q(\mathbf{u})d\mathbf{u}$, where $p_\theta(\mathbf{f}|\mathbf{u})$ is a Gaussian density according to the GP prior of the DKL, and $q(\mathbf{u})$ is a variational Gaussian distribution with learned parameters. We note that while we omit $\mathbf{Z}$ for brevity, it plays an important role as the kernel matrix depends on $\mathbf{Z}$. Now we can plug this posterior distribution in Eq. 4:

$$\mathbb{E}_{\mathcal{D}_1,\mathcal{D}_2}\mathbb{E}_{\mathbf{x}_*,y_*\sim\mathcal{D}_2} D_{KL}[q_\theta(f_*|\mathbf{x}_*,\mathbf{Z})||p(f_*|\mathbf{x}_*,y_*,\mathcal{D}_1)], \tag{8}$$

and obtain the objective in Eq. 7 with the new posterior:

$$\begin{aligned}&\mathbb{E}_{\mathcal{D}_1,\mathcal{D}_2}\mathbb{E}_{\mathbf{x}_*,y_*\sim\mathcal{D}_2}\{\mathbb{E}_{q_\theta(f_*|\mathbf{x}_*,\mathbf{Z})}[-\log\ p(y_*|f_*)]\\&+\beta\cdot D_{KL}[q_\theta(f_*|\mathbf{x}_*,\mathbf{Z})||p(f_*|\mathbf{x}_*,\mathcal{D}_1)]\}.\end{aligned} \tag{9}$$

A key part of scaling our objective is to allow for mini-batching. The objective in Eq. 9 naturally factorizes over the data points in $\mathcal{D}_2$, which leaves the $\mathcal{D}_1$ terms. As we split our dataset at each iteration into a train and validation set, a simple solution is to split a random batch instead of the entire dataset. Namely, given a batch of examples $\mathcal{B}$ we split it to two subsets $\mathcal{B}_1$ and $\mathcal{B}_2$ similar to the split we did for $\mathcal{D}$ and compute the following objective:

$$\begin{aligned}&\mathbb{E}_{\mathcal{B}_1,\mathcal{B}_2}\mathbb{E}_{\mathbf{x}_*,y_*\sim\mathcal{B}_2}\{\mathbb{E}_{q_\theta(f_*|\mathbf{x}_*,\mathbf{Z})}[-\log\ p(y_*|f_*)]\\&+\beta\cdot D_{KL}[q_\theta(f_*|\mathbf{x}_*,\mathbf{Z})||p(f_*|\mathbf{x}_*,\mathcal{B}_1)]\}.\end{aligned} \tag{10}$$

In this case, the inducing locations $\mathbf{Z}$ and the variational parameters of $q(\mathbf{u})$ are also learned as part of the optimization process. Nevertheless, we achieve two important goals. First, the DKL model still tries to match the posterior of the label-informed NNGP model. Second, we need to evaluate the NNGP model only on the actual data points which are known in advance and can be computed beforehand. Thus, we are avoiding costly evaluations on the inducing inputs by this model. Furthermore, we can define the inducing inputs in the feature space of the NN which is much more beneficial in terms of optimization compared to the input space of the network [Bradshaw et al., 2017, Achituve et al., 2021a]. As a byproduct, we obtained a straightforward method to incorporate the inducing points mechanism in GP models that belong to the class of infinite-width networks.

4 RELATED WORK

Bayesian NNs. Bayesian NNs model the uncertainty over the true underline function by assuming a probability distribution over the network parameters [Minka, 2000]. Instead of solving an optimizing process for a single set of parameters, BNNs attempt to compute the Bayesian model average (BMA) [Wilson and Izmailov, 2020]. However, for modern NNs solving the BMA integral is computationally intractable and approximations must be used. Notable examples are the Laplace approximation [MacKay, 1992, Khan et al., 2019, Daxberger et al., 2021], MCMC-based methods [Neal, 1996, Welling and Teh, 2011, Chen et al., 2014, Zhang et al., 2020], and variational inference [Graves, 2011, Blundell et al., 2015, Kingma et al., 2015]. These approximations usually result in either degraded performance, specialized NN architectures, unreliable uncertainty estimation, or computational difficulties in terms of memory and time. One possible compelling alternative for making inference in parameter space is to do it in function spaces [Sun et al., 2019, Wang et al., 2019, Rudner et al., 2021, Ma and Hernández-Lobato, 2021]. However, these methods suffer from similar issues as standard BNNs do and may involve rough approximations. A different line of work considers the distribution over functions when using infinite-width layers in fully connected networks [Neal, 1996, Lee et al., 2018, de G. Matthews et al., 2018, Jacot et al., 2018]. This approach allows performing tractable Bayesian inference with NNs while avoiding the optimization difficulties associated with training them. Later, this approach was extended to other architectures and layers, such as CNNs, RNNs, attention, and batch normalization [Garriga-Alonso et al., 2019, Novak et al., 2019, Yang, 2019]. However, these approaches suffer from several drawbacks, such as reduced generalization, and costly kernel evaluation. Lastly, there exists some evidence [Brosse et al., 2020, Kristiadi et al., 2020] that being Bayesian, even only on the last layer, can provide the desirable benefits of the Bayesian paradigm.

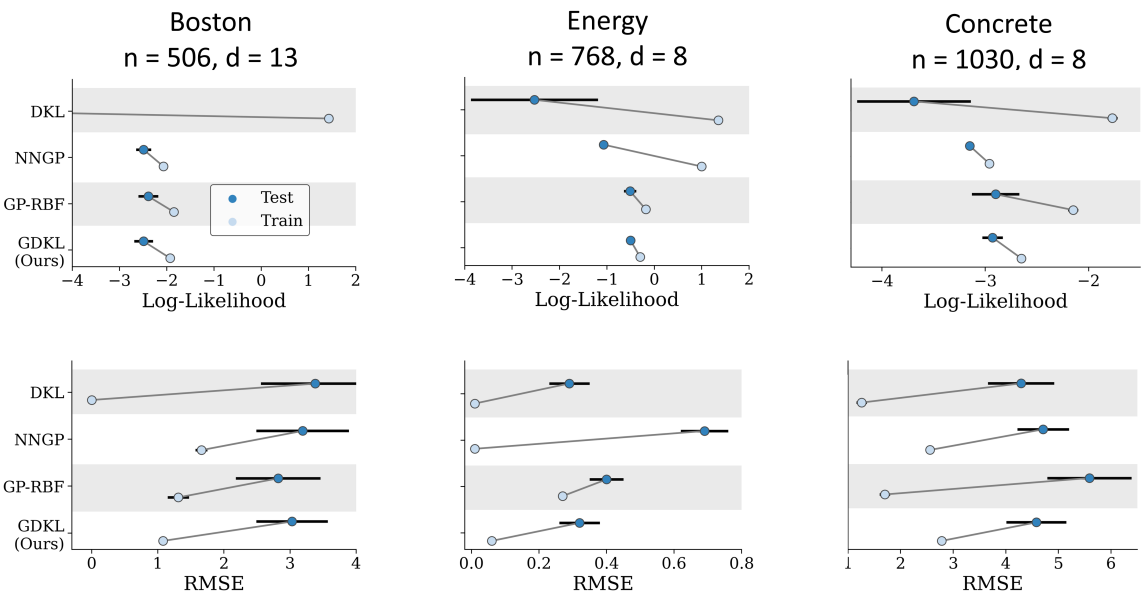

Figure 2: Results for small UCI datasets. We report the log-likelihood (top; right is better) and RMSE (bottom; left is better) for each method over 10 splits on the training test sets. The log-likelihood of DKL on Boston is ∼-400.

Learning representations with GPs. An alternative approach for learning in function spaces is with Gaussian processes. However, Gaussian processes cannot learn a new representation of the data [Wilson et al., 2016a]. Effectively this limits them to data modalities on which standard kernels can capture similarity well. Two common solutions for this problem are deep GPs [Damianou and Lawrence, 2013, Salimbeni and Deisenroth, 2017] and deep kernel learning [Calandra et al., 2016, Wilson et al., 2016a]. In this study, we build on the latter approach. Unfortunately, it was found that DKLs can overfit in a particular way [Ober et al., 2021]. The DKL objective will tend to correlate all data points instead of only those that convey information about each other. One way to mitigate this phenomenon is to use a fully Bayesian approach [Ober et al., 2021]. Yet, this direction inherits the challenges of working with BNNs which one would like to avoid when using DKLs. Simultaneously, several studies suggested methods to tackle this limitation of DKLs. van Amersfoort et al. [2021] proposed to do spectral normalization to the NNs parameters in architectures with residual connections following [Liu et al., 2020]. However, this method is limited to networks with residual connections only and may depend heavily on the estimation quality of the spectral norm. Also, in our experiments we often found this method to be equivalent to standard DKLs. Liu et al. [2021] proposed to use stochastic NNs to learn the representations of examples. The first method, termed DLVKL, uses an encoder network, similar to that used in VAEs [Kingma and Welling, 2014]. The second method, termed DLVKL-NSDE, uses stochastic differential equation flows. To use flow-based models, the feature and input spaces must have the same dimensionality which makes this method impractical for high-dimensional data. Lastly, Mallick et al. [2021] proposed to map data points to probability distributions using probabilistic NNs and fit a GP in that space. This method builds on particle-based optimization [Liu and Wang, 2016] and as such it operates on several NNs simultaneously which may be challenging for even moderate-sized NNs. We would like to note, that we expect our method to gain benefit from similar techniques (e.g., [Lakshminarayanan et al., 2017]). We leave this direction to future research endeavors.

5 EXPERIMENTS

We evaluated GDKL on a number of benchmark regression and classification datasets, ranging from small to medium size, and low to high data dimensionality. Unless stated otherwise, in all experiments we report the mean performance (e.g., log-likelihood) along with one standard deviation over random seeds, which may include randomness in the data and the parameters. We stress that we used the same initialization for all compared methods. Full implementation details are given in Appendix A, and further experiments are presented in Appendix C. For reproducibility, code will be published as well.

5.1 SMALL-SIZED DATASETS

To showcase our claim that GDKL can learn from small datasets while being robust to overfitting, we first evaluated it on the three small-sized UCI benchmark datasets [Dua and Graff, 2017]: Boston, Energy, and Concrete. We compared the exact GP variant of our method to (1) *DKL* - standard

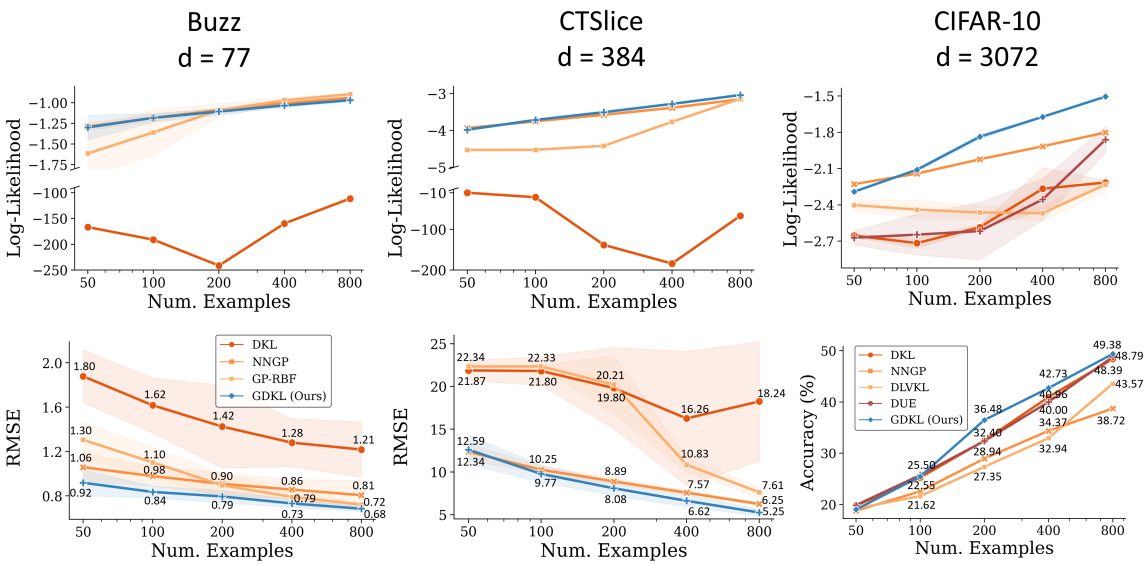

Figure 3: Model performance on $[50, 100, 200, 400, 800]$ training data points (x-axis in log-scale). We report the log-likelihood (top) and RMSE/accuracy (bottom) on the test set for all datasets. A higher log-likelihood and accuracy, and a lower RMSE are better. All the results are based on three random seeds. On Buzz and CTSlice we didn't report here the standard deviation of the log-likelihood for the DKL model as it was very large and impaired the visibility of the figure.

DKL training [Calandra et al., 2016, Wilson et al., 2016a]; (2) *NNGP* - A GP with an NNGP kernel [Lee et al., 2018, de G. Matthews et al., 2018]; and (3) *GP-RBF* - Standard GP with an RBF kernel (without DKL). The last simple baseline is considered a strong approach on these datasets [Salimbeni and Deisenroth, 2017], as the semantic similarity is well captured by the RBF kernel. All neural network models (NNGP, DKL, and GDKL) use a three hidden layer fully-connected network. DKL and GDKL use the same width for each layer. We follow the training protocol suggested in [Ober et al., 2021] with several modifications which are described in Appendix A. As customary on these datasets (e.g., [Salimbeni and Deisenroth, 2017]), we use k-fold cross validation with 90% randomly selected data as training and the remaining 10% as a held-out test set. Here we used $k = 10$. We scale the inputs and outputs of each partition of the data to have zero mean and unit standard deviation based on the training part only (the output scaling is restored in evaluation). Figure 2 shows the Log-Likelihood (LL) and RMSE of the compared methods.

From the figure, we observe several findings. First, we indeed observe that standard DKL training produces a model that has the characteristics of a NN and not a Bayesian model, i.e. it has good RMSE values at train and test, but it does not reliably estimate its uncertainty, as seen from its inferior test log-likelihood values. Second, the RMSE performance of NNGP on the Energy dataset is considerably worse than other baselines, confirming our claim that NNGP can have poor predictive performance. Finally, GDKL is always comparable to the best of the two on all datasets on both metrics and is substantially less prone to overfitting.

5.2 HIGH-DIMENSIONAL DATASETS

Next, we expect to achieve the most benefit from DKL in settings with high dimensional data, where standard kernels do not perform as well. In these cases, the NN should be encouraged to find a low-dimensional representation over which a GP will work well. To test that scenario we considered the two regression datasets Buzz and CTSlice from the UCI repository, and the classification dataset CIFAR-10 [Krizhevsky et al., 2009]. Here, we use a subset of the training data with a varied number of training examples from 50 to 800 and recorded the log-likelihood and the RMSE/accuracy of the model on the test set in each experiment. For Buzz and CTSlice we allocated 10% of the data for testing, and for CIFAR-10 we use the default test split. On the regression datasets, we compared the exact GP variant of our method to the same baselines as in Section 5.1. On CIFAR-10, we use the Dirichlet-based likelihood function suggested in [Milios et al., 2018] for inference. On this dataset, we didn't compare to the GP-RBF baseline as it works poorly on images. However, we did compare to two additional baselines: (1) *DLVKL* [Liu et al., 2021] which learns a stochastic encoder network, reminiscent of VAEs [Kingma and Welling, 2014], to promote a regularized representation of the data; and (2) *DUE* [van Amersfoort et al., 2021] which applies spectral normalization on the weights

Table 1: Test results on full CIFAR-10 and CIFAR-100 based on three random seeds.

	CIFAR-10				CIFAR-100			
	ACC (↑)	LL (↑)	ECE (↓)	MCE (↓)	ACC (↑)	LL (↑)	ECE (↓)	MCE (↓)
DKL	95.45 ± 0.06	-0.19 ± 0.00	0.03 ± 0.00	0.30 ± 0.01	77.90 ± 0.45	-0.94 ± 0.00	0.08 ± 0.00	0.24 ± 0.02
DLVKL	$\mathbf{95.65 \pm 0.12}$	-0.18 ± 0.00	0.03 ± 0.00	0.42 ± 0.20	77.42 ± 0.05	-0.95 ± 0.02	0.09 ± 0.00	0.26 ± 0.02
DUE	95.48 ± 0.09	-0.19 ± 0.00	0.03 ± 0.00	0.32 ± 0.04	76.39 ± 0.23	-0.98 ± 0.03	0.10 ± 0.01	$\mathbf{0.16 \pm 0.02}$
GDKL (Ours)	$\mathbf{95.67 \pm 0.06}$	$\mathbf{-0.17 \pm 0.00}$	$\mathbf{0.01 \pm 0.00}$	$\mathbf{0.27 \pm 0.01}$	$\mathbf{78.36 \pm 0.19}$	$\mathbf{-0.89 \pm 0.01}$	$\mathbf{0.06 \pm 0.00}$	0.18 ± 0.01

with architectures that contain residual connections. Note that unlike GDKL these baselines require some modification to the NN. Here, we used a variant of the wide residual network (WRN) [Zagoruyko and Komodakis, 2016] as a feature extractor in DKL models. As for the NNGP baseline (and for modeling p in GDKL), we used a variant of this network without the average pooling layer as it imposes a large computational burden. The results on the three datasets are shown in Figure 3.

From the figures, we observe again that the DKL model overfits strongly, and in some cases, even its mean prediction is substantially lower than baseline methods. In addition, the NNGP works well on the regression datasets, but less so on real images. And finally, here as well, across all training set sizes, GDKL achieves the highest, or comparable, results in both the log-likelihood and RMSE/accuracy. In Appendix C we also quantify the uncertainty of the models through calibration on the CIFAR-10 dataset. We compare all methods both visually using reliability diagrams and common metrics [Brier, 1950, Guo et al., 2017] on all dataset sizes. The figures show that GDKL is best calibrated across all metrics in all cases when $n \geq 200$, and on smaller dataset sizes it is second only to the NNGP model.

5.3 MEDIUM-SIZED, HIGH-DIMENSIONAL DATASETS

Having established that GDKL works well in low-date regime settings, we now evaluate its performance on larger datasets in which exact inference is more challenging. We do so on the full CIFAR-10 and CIFAR-100 datasets. We compare GDKL to the standard DKL baseline, and to DLVKL and DUE which were presented in Section 5.2. In general, we followed the protocol suggested in [van Amersfoort et al., 2021] for training on the CIFAR-10 dataset having only 10 inducing points. For CIFAR-100 we used a similar protocol with 200 inducing points. Exact experimental details are given in Appendix A. Here, as well, we used a variant of the WRN for both the DKL models and the NNGP model used by GDKL. Table 1 shows the test accuracy, log-likelihood, expected calibration error (ECE), and maximum calibration error (MCE) for both datasets. The ECE measures a weighted average distance between the classifier's confidence and accuracy, and the MCE measures the maximum instead of the average. From the table, GDKL outperforms all baselines in almost all of the cases. Note how GDKL is able to maintain and even surpass the accuracy of DKL while providing a classifier that is better calibrated.

6 CONCLUSIONS

In this study, we put forward a novel method for learning deep kernels. Our goal is to train deep kernels that keep the benefits of Bayesian models without sacrificing performance. To this end, we define a new training procedure that uses an infinite-width NN to guide the DKL optimization, effectively setting adaptive levels of confidence in our predictions. This objective utilizes the reliable uncertainty estimation of NNGPs to allow our model to be as confident as possible without being over-confident. Finally, we also proposed an extension of our model to incorporate inducing points. We evaluated GDKL on small to mid-sized datasets having low and high data dimensionality. We found that our method consistently generalized well to novel data points while not scarifying the Bayesian properties of it, i.e., it doesn't overfit. As a possible future research direction, it would be interesting to combine our framework with Bayesian models other than infinite-width NNs.

References

I. Achituve, A. Navon, Y. Yemini, G. Chechik, and E. Fetaya. GP-Tree: A Gaussian process classifier for few-shot incremental learning. In *International Conference on Machine Learning*, pages 54–65. PMLR, 2021a.

I. Achituve, A. Shamsian, A. Navon, G. Chechik, and E. Fetaya. Personalized federated learning with Gaussian processes. *Advances in Neural Information Processing Systems*, 34:8392–8406, 2021b.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

J. Bradshaw, A. G. d. G. Matthews, and Z. Ghahramani. Adversarial examples, uncertainty, and transfer testing robustness in Gaussian process hybrid deep networks. *arXiv preprint arXiv:1707.02476*, 2017.

G. W. Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

N. Brosse, C. Riquelme, A. Martin, S. Gelly, and É. Moulines. On last-layer algorithms for classification: Decoupling representation from uncertainty estimation. *arXiv preprint arXiv:2001.08049*, 2020.

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

T. Chen, E. Fox, and C. Guestrin. Stochastic gradient Hamiltonian Monte Carlo. In *International conference on machine learning*, pages 1683–1691. PMLR, 2014.

Y. Cho and L. Saul. Kernel methods for deep learning. *Advances in neural information processing systems*, 22, 2009.

A. Damianou and N. D. Lawrence. Deep Gaussian processes. In *Artificial intelligence and statistics*, pages 207–215. PMLR, 2013.

E. Daxberger, A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, and P. Hennig. Laplace redux-effortless Bayesian deep learning. *Advances in Neural Information Processing Systems*, 34:20089–20103, 2021.

A. G. de G. Matthews, J. Hron, M. Rowland, R. E. Turner, and Z. Ghahramani. Gaussian process behaviour in wide deep neural networks. In *6th International Conference on Learning Representations, ICLR*, 2018.

D. Dua and C. Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

A. Garriga-Alonso, C. E. Rasmussen, and L. Aitchison. Deep convolutional networks as shallow Gaussian processes. In *7th International Conference on Learning Representations, ICLR*, 2019.

A. Graves. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24, 2011.

C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In *Uncertainty in Artificial Intelligence*, page 282. Citeseer, 2013.

A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

M. Jankowiak, G. Pleiss, and J. Gardner. Parametric Gaussian process regressors. In *International Conference on Machine Learning*, pages 4702–4712. PMLR, 2020.

M. E. E. Khan, A. Immer, E. Abedi, and M. Korzepa. Approximate inference turns deep networks into Gaussian processes. *Advances in neural information processing systems*, 32, 2019.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR*, 2014.

D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28, 2015.

A. Kristiadi, M. Hein, and P. Hennig. Being Bayesian, even just a bit, fixes overconfidence in Relu networks. In *International conference on machine learning*, pages 5436–5446. PMLR, 2020.

A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. Deep neural networks as Gaussian processes. In *6th International Conference on Learning Representations, ICLR*, 2018.

S. W. Linderman, M. J. Johnson, and R. P. Adams. Dependent multinomial models made easy: stick breaking with the Pólya-Gamma augmentation. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pages 3456–3464, 2015.

H. Liu, Y.-S. Ong, X. Jiang, and X. Wang. Deep latent-variable kernel learning. *IEEE Transactions on Cybernetics*, 52(10):10276–10289, 2021.

J. Liu, Z. Lin, S. Padhy, D. Tran, T. Bedrax Weiss, and B. Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *Advances in Neural Information Processing Systems*, 33:7498–7512, 2020.

Q. Liu and D. Wang. Stein variational gradient descent: A general purpose Bayesian inference algorithm. *Advances in neural information processing systems*, 29, 2016.

S. Lotfi, P. Izmailov, G. Benton, M. Goldblum, and A. G. Wilson. Bayesian model selection, the marginal likelihood, and generalization. In *International Conference on Machine Learning*, pages 14223–14247. PMLR, 2022.

C. Ma and J. M. Hernández-Lobato. Functional variational inference based on stochastic process generators. *Advances in Neural Information Processing Systems*, 34: 21795–21807, 2021.

D. J. MacKay. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992.

A. Mallick, C. Dwivedi, B. Kailkhura, G. Joshi, and T. Y.-J. Han. Deep kernels with probabilistic embeddings for small-data learning. In *Uncertainty in Artificial Intelligence*, pages 918–928. PMLR, 2021.

D. Milios, R. Camoriano, P. Michiardi, L. Rosasco, and M. Filippone. Dirichlet-based Gaussian processes for large-scale calibrated classification. *Advances in Neural Information Processing Systems*, 31, 2018.

T. P. Minka. Bayesian model averaging is not model combination. *Available electronically at http://www. stat. cmu. edu/minka/papers/bma. html*, pages 1–2, 2000.

R. M. Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 1996.

R. Novak, L. Xiao, Y. Bahri, J. Lee, G. Yang, J. Hron, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein. Bayesian deep convolutional networks with many channels are Gaussian processes. In *7th International Conference on Learning Representations, ICLR*, 2019.

R. Novak, L. Xiao, J. Hron, J. Lee, A. A. Alemi, J. Sohl-Dickstein, and S. S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *8th International Conference on Learning Representations, ICLR*, 2020.

S. W. Ober, C. E. Rasmussen, and M. van der Wilk. The promises and pitfalls of deep kernel learning. In *Uncertainty in Artificial Intelligence*, pages 1206–1216. PMLR, 2021.

N. G. Polson, J. G. Scott, and J. Windle. Bayesian inference for logistic models using Pólya–Gamma latent variables. *Journal of the American Statistical Association*, pages 1339–1349, 2013.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

T. G. Rudner, Z. Chen, Y. W. Teh, and Y. Gal. Tractable function-space variational inference in bayesian neural networks. In *Advances in Neural Information Processing Systems*, 2021.

H. Salimbeni and M. Deisenroth. Doubly stochastic variational inference for deep Gaussian processes. *Advances in neural information processing systems*, 30, 2017.

J. Snell and R. Zemel. Bayesian few-shot classification with one-vs-each Pólya-Gamma augmented Gaussian processes. In *International Conference on Learning Representations*, 2021.

S. Sun, G. Zhang, J. Shi, and R. Grosse. Functional variational Bayesian neural networks. In *7th International Conference on Learning Representations, ICLR*, 2019.

M. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial intelligence and statistics*, pages 567–574. PMLR, 2009.

J. van Amersfoort, L. Smith, A. Jesson, O. Key, and Y. Gal. On feature collapse and deep kernel learning for single forward pass uncertainty. *arXiv preprint arXiv:2102.11409*, 2021.

G. Vardi and O. Shamir. Implicit regularization in relu networks with the square loss. In *Conference on Learning Theory (COLT)*, 2021.

Z. Wang, T. Ren, J. Zhu, and B. Zhang. Function space particle optimization for Bayesian neural networks. In *7th International Conference on Learning Representations, ICLR*, 2019.

M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.

C. Williams. Computing with infinite networks. *Advances in neural information processing systems*, 9, 1996.

A. G. Wilson and P. Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *Advances in neural information processing systems*, 33:4697–4708, 2020.

A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016a.

A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Stochastic variational deep kernel learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2594–2602, 2016b.

G. Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019.

S. Zagoruyko and N. Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference BMVC*, 2016.

R. Zhang, C. Li, J. Zhang, C. Chen, and A. G. Wilson. Cyclical stochastic gradient MCMC for Bayesian deep learning. In *8th International Conference on Learning Representations, ICLR*, 2020.