Scalable Uncertainty for Computer Vision with Functional Variational Inference

Eduardo D C Carvalho* Ronald Clark* Andrea Nicastro* Paul H J Kelly*

Abstract

As Deep Learning continues to yield successful applications in Computer Vision, the ability to quantify all forms of uncertainty is a paramount requirement for its safe and reliable deployment in the real-world. In this work, we leverage the formulation of variational inference in function space, where we associate Gaussian Processes (GPs) to both Bayesian CNN priors and variational family. Since GPs are fully determined by their mean and covariance functions, we are able to obtain predictive uncertainty estimates at the cost of a single forward pass through any chosen CNN architecture and for any supervised learning task. By leveraging the structure of the induced covariance matrices, we propose numerically efficient algorithms which enable fast training in the context of high-dimensional tasks such as depth estimation and semantic segmentation. Additionally, we provide sufficient conditions for constructing regression loss functions whose probabilistic counterparts are compatible with aleatoric uncertainty quantification.

1. Introduction

Supervised learning, in its deterministic formulation, involves learning a mapping $f : \mathcal{X} \to \mathcal{Y}$ given observed data $\mathcal{D}_N = \{x_i, y_i\}_{i=1}^N = \{\boldsymbol{X}_D, \boldsymbol{y}_D\}$. In a Deep Learning context, f is parametrized by a neural network whose architecture expresses convenient inductive biases for the task of interest and whose training consists on optimizing a loss function with respect to its parameters by using stochastic optimization techniques. Despite its widespread empirical success, Deep Learning approaches are hardly ever transparent, so that in certain domains, such as medical diagnosis or self-driving vehicles, it becomes unclear how to map predictions on unseen inputs to a non-catastrophic decision. Thus much research has been focused on obtaining uncertainties from deep models for common computer vision tasks such as semantic segmentation [18, 16, 33], depth estimation [20, 24], visual odometry [2, 46, 7, 6], SLAM [8] and active learning [10].

A more reliable approach is to consider a Bayesian probabilistic formulation of deep supervised learning, also known as Bayesian Deep Learning [32, 34], so that all forms of predictive uncertainty may be quantified. There are two types of uncertainty one may encounter: *epistemic* and *aleatoric* [20], both which are naturally accounted for in a Bayesian framework. Epistemic uncertainty is associated with a model's inability of finding a meaningful mapping from inputs to outputs and will eventually vanish as it is trained on a large and diverse dataset. Epistemic uncertainty becomes particularly relevant when the trained model has to make predictions on input examples which, in some sense, differ significantly from training data: out-of-distribution (OOD) inputs [13]. Aleatoric uncertainty is associated to noise contained in the observed data and cannot be reduced as more data is observed, nor does it increase on OOD inputs, so that it is not able to detect these by itself. Modelling the combination of epistemic and aleatoric uncertainties is therefore key in order to build deep learning based systems which are transparent about their predictive capabilities.

1.1. General background

Denoting all parameters of a neural network as W, Bayesian Deep Learning starts with positing a prior distribution $\pi(W)$, typically multivariate normal, and a likelihood $p(y|T(x;W))$, where $T(.;W)$ is a neural network with weights W. The solution to this bayesian inference problem is the posterior over weights $p(W|\mathcal{D}_N)$, which is unknown due to the intractable computation of marginal likelihood $p(\mathcal{D}_N)$. Stochastic variational inference (SVI) [12, 15] allows one to perform scalable approximate posterior inference, hence being the dominant paradigm in Bayesian Deep Learning. Denoting $q(W)$ as the variational distribution and $\mathcal{D}_B$ as a mini-batch of size B, the following training objective is considered:

$$\frac{N}{B}\sum_{i=1}^{B}\mathbb{E}_{q(W)}\left[\log p(y_i|T(x_i;W))\right] - \mathrm{KL}\left(q(W)||\pi(W)\right) \tag{1}$$

This quantity is denoted as evidence lower bound (ELBO), given that it is bounded above by $\log p(\mathcal{D}_N)$. By choosing a convenient family of distributions for $q(W)$ and

*Authors are with Department of Computing, Imperial College London. Correspondence to eduardo.carvalho16@ic.ac.uk or ronald.clark@ic.ac.uk

suitably parametrizing it with neural network mappings, approximate bayesian inference amounts to maximizing the ELBO with respect to its parameters over multiple minibatches $\mathcal{D}_B$. The success of variational inference (VI) depends on the expressive capability of $q(W)$, which ideally should be enough to approximate $p(W|\mathcal{D}_N)$. Even though considerable work has been done in designing various variational families for BNN posterior inference [4, 29, 30, 42], these are not easily applicable in computer vision tasks which require large network architectures.

Alternatively, a nonparametric formulation of probabilistic supervised learning is obtained by introducing a stochastic process over a chosen function space. An $\mathcal{F}$ valued stochastic process with index set $\mathcal{X}$ is a collection of random variables $\{f(x)\}_{x\in\mathcal{X}}$ whose distribution is fully determined by its finite n-dimensional marginal distributions $p(f^{\boldsymbol{X}})$, for any $\boldsymbol{X} = (x_1, ..., x_n) \in \mathcal{X}^n$, $n \in \mathbb{N}$, and where $f^{\boldsymbol{X}} = (f(x_1), ..., f(x_n))$. An important class are Gaussian Processes (GPs) [39], which are defined by a mean function $m(.)$ and covariance kernel $k(.,.)$, and all its finite dimensional marginal distributions are multivariate gaussians: $p(f^{\boldsymbol{X}}) = \mathcal{N}(m(\boldsymbol{X}), k(\boldsymbol{X}, \boldsymbol{X}))$, where $m(\boldsymbol{X})$ is a mean vector and $k(\boldsymbol{X}, \boldsymbol{X})$ a covariance matrix.

Bayesian Neural Networks (BNNs) may also be viewed as prior distributions over functions by means of a two-step generative process. Firstly one draws a prior sample $W \sim \pi(W)$, and then a single function is defined by setting $f(.) = T(.; W)$. BNNs are an example of implicit stochastic processes [31], where for any finite set of inputs $\boldsymbol{X}$ its distribution may be written as follows:

$$p\left(f^{\boldsymbol{X}} \in A\right) = \int_{\{T(\boldsymbol{X};W)=f^{\boldsymbol{X}}\in A\}} \pi(W)dW \qquad (2)$$

Where $p(.)$ is a probability measure and A is an arbitrary measurable set. Even though it is easy to sample from $p(.)$, it is not generally possible to exactly compute its value due to non-invertibility of $T(.; W)$. Note that in this formulation the dimensionality of the BNN prior does not depend on the dimensionality of weight space, meaning that posterior inference over a BNN with millions of weights only depends on the number of inputs n and dimensionality of $\mathcal{F}$, which is significantly smaller. Moreover, while $p(W|\mathcal{D}_N)$ may have complex structure due to the fact that many different values of W yield the same output values, this can largely be avoided if one performs VI directly in function space [31].

1.2. List of contributions

Our contributions are the following:

1. Given any loss function of interest for regression tasks, we provide sufficient conditions for constructing well-defined likelihoods which are compatible with aleatoric uncertainty quantification, and provide a practically relevant example based on the reverse Huber loss [26, 25].

2. Leveraging the functional VI framework from [44], we propose a computationally scalable variant which uses a suitably parametrized GP as the variational family. Following [11], we are able to associate certain Bayesian CNN priors with a closed-form covariance kernel, which we then use to define a GP prior. Assuming the prior is independent across its output dimensions, we propose an efficient method for obtaining its inverse covariance matrix and determinant, hence allowing functional VI to scale to high-dimensional supervised learning tasks. After training, this constitutes a practically useful means of obtaining predictive uncertainty (both epistemic and aleatoric) at the cost of a single forward pass through the network architecture, hence opening new directions for encompassing uncertainty quantification into real-time prediction tasks [20].

3. We apply this approach in the context of semantic segmentation and depth estimation, where we show it displays well-calibrated uncertainty estimates and error metrics which are comparable with other approaches based on weight-space VI objectives.

2. Functional Variational Inference

2.1. Background

Even though GPs offer a principled way of handling uncertainty in supervised learning, performing exact inference carries a cubic cost in the number of data points, thus preventing its applicability to large and high-dimensional datasets. Sparse variational methods [45, 14] overcome this issue by allowing one to compute variational posterior approximations using subsets of training data, but it is difficult to choose an appropriate set of inducing points in the context of image-based datasets [41].

Functional Variational Bayesian Neural Networks (FVBNNs) [44] use BNNs to approximate function posteriors at finite sets of inputs. This is made possible by defining a KL divergence on general stochastic processes (see [44] for the definition and proof). Building upon such divergence, and defining $\boldsymbol{X'} \in \mathcal{X}^{n'}$, where n' is fixed, and setting $\boldsymbol{X} = \boldsymbol{X}_D \cup \boldsymbol{X'}$, it is possible to obtain a practically useful analogue of ELBO in function space:

$$\sum_{i=1}^{N} \mathbb{E}_{q(f(x_i))}\left[\log p(y_i|f(x_i))\right] - \mathrm{KL}\left(q(f^{\boldsymbol{X}})||p(f^{\boldsymbol{X}})\right) \qquad (3)$$

We refer to this equation as the *functional VI* objective, whose structure will be discussed and simplified during the

next sections in order to yield a more computationally feasible version which does not use BNNs as the variational family nor does so explicitly for its prior.

This objective is valid since it is bounded above by $\log p(\mathcal{D}_N)$ for any choice of $\boldsymbol{X'}$ [44]. In practice $\mathcal{D}_N$ is replaced by an expectation over a mini-batch $\mathcal{D}_B$, so that the corresponding ELBO is only a lower-bound to $\log p(\mathcal{D}_B)$ and not $\log p(\mathcal{D}_N)$. During training $\boldsymbol{X'}$ may be sampled at random in order to cover the input domain, such as adding gaussian noise to the existing training inputs. Whenever $\boldsymbol{X'}$ are far from training inputs, $q(.)$ will be encouraged to fit the prior process, whereas the data-driven term will dominate on input locations closer to training data. In this way, the question of obtaining reliable predictive uncertainty estimates on OOD inputs gets reduced to choosing a meaningful prior distribution over functions. In this work we will be choosing $p(.)$ to be Bayesian CNNs, which constitute a diverse class of function priors on image space.

2.2. Logit attenuation for classification in functional VI

We now consider classification tasks under the functional VI objective (3), where we assume that $\mathcal{Y} = \{0, 1\}^K$, K is the number of distinct classes and $\mathcal{F} = \mathbb{R}^K$. One of the limitations of this objective is that it is not a lower bound to the log-marginal likelihood of the training dataset. When the true function posterior is not in the same class as $q(.)$, there is no guarantee that this procedure will provide reasonable results [41]. We have observed this when we have first tried it in our segmentation experiments, which has caused model training to converge very slowly.

In order to mitigate this issue, we consider the following discrete likelihood under the functional VI framework:

$$p(y_k|f(x)) = \frac{\exp\left(f_k^{'}(x)\right)}{\sum_{k=1}^{K}\exp\left(f_k^{'}(x)\right)} \tag{4}$$

Where $f_k^{'}(x) = f_k(x)/\sigma_k^2(x)$, so that $p(y_k|f(x))$ is a Boltzmann distribution with re-scaled logits, where scale parameter $\sigma_k^2(x)$ weighs its corresponding logit $f_k(x)$. When included into the functional VI objective (3), this parametrization enables the model to become robust to erroneous class labels contained in the training data, while also avoiding over-regularization from the function prior which may lead to underfitting. This effect of logit attenuation naturally yields a change in aleatoric uncertainty, as measured in entropy. Moreover, we note that each $\sigma_k^2(x)$ is not easily interpretable in terms of inducing higher or smaller aleatoric uncertainty according to its respective magnitude, so that one has to rely on measuring the total predictive uncertainty in terms of the predictive entropy. Additionally, when encompassed into deterministic models or the weight-space ELBO in (1), re-scaling logits brings no added flexibility.

3. Functional VI with general regression loss functions

It is often the case that best-performing non-probabilistic approaches in computer vision tasks not only have carefully crafted network architectures, but also task-specific loss functions which allow one to encode relevant inductive biases. The most standard examples are the correspondence between gaussian likelihood and $\mathcal{L}_2$ loss, and also between laplacian likelihood and $\mathcal{L}_1$. However, various loss functions of interest are not immediately recognized as being induced by a known probability distribution, so that it would be of practical relevance to start with positing a loss function and then derive its corresponding likelihood model. Given any additive loss function $\ell : \mathcal{Y} \times \mathcal{F} \to \mathbb{R}_{\geq 0}$, we define its associated likelihood as follows:

$$p(y|f(x)) = \frac{\exp\left(-\ell(y, f(x))\right)}{Z} \tag{5}$$

This is known as the Gibbs distribution with energy function ℓ and temperature parameter set to 1. $Z = \int_{\mathcal{Y}} \exp\left(-\ell(y, f(x))\right) dy$ is its normalization constant, potentially depending on $f(x)$, which can either be computed analytically or using numerical integration. Any loss function $\ell(.,.)$ for which Z is finite can be made into a likelihood model, hence being consistent with Bayesian reasoning. Moreover, any strictly positive probability density can be represented as in (5) for some appropriate choice of ℓ, which follows from the Hammersley-Clifford theorem [1]. In the context of computer vision, typically involving large amounts of labelled and noise-corrupted data, aleatoric uncertainty tends to be the dominant component of predictive uncertainty [20]. This means that, for each task of interest, one needs to restrict from choosing arbitrary likelihoods to the ones which are compatible with modelling this type of uncertainty. In the following subsection we provide a means of doing so for the task of regression.

3.1. Aleatoric uncertainty for regression

Without loss of generality, we assume that $\mathcal{Y} = \mathcal{F} = \mathbb{R}$, so that $p(y|f(x))$ is a univariate conditional density. This covers most practical cases of interest, including per-pixel regression tasks such as depth estimation, and simplifies the notation considerably.

In regression tasks, we are typically interested in writing loss functions of the form $\ell(y, f(x)) = \ell\left(\frac{y-f(x)}{\sigma(x)}\right)$, where $f(x)$ and $\sigma(x)$ are location and scale parameters, respectively. Writing $\ell(y)$ as the standardized loss, we define the standard member of its family of Gibbs distributions as $p_0(y) = \frac{1}{Z_0}\exp(-\ell(y))$. Then $p(y|f(x)) = \frac{1}{Z}\exp\left(-\ell\left(\frac{y-f(x)}{\sigma(x)}\right)\right)$, where $Z = \sigma(x)Z_0$, defines a valid location-scale family of likelihoods. Moreover, we require

its first and second moments to be finite, so that we may compute or approximate means and variances of the predictive distribution. For instance, this excludes using the Cauchy distribution as a likelihood. Substituting into equation 3 and ignoring additive constants, we obtain the following training objective:

$$-\sum_{i=1}^{n}\left(\mathbb{E}_{q(f(x_i))}\left[\ell\left(\frac{y_i-f(x_i)}{\sigma(x_i)}\right)\right]+\log\left(\sigma(x_i)\right)\right) - \mathrm{KL}\left(q(f^{\boldsymbol{X}})||p(f^{\boldsymbol{X}})\right) \quad (6)$$

Similarly to [20, 21], we interpret each $\sigma(x_i)$ as a loss attenuation factor which may be learned during training and $\log(\sigma(x_i))$ as its regularization component.

In order to display the practical utility of this loss-based construction, we consider the reverse Huber (berHu) loss from [26], which has previously been considered in [25] for improving monocular depth estimation, and derive its probabilistic counterpart, which we denote as berHu likelihood (see supplementary material).

4. Scaling Functional VI to high-dimensional tasks

Various priors of interest in computer vision applications, including Bayesian CNNs, are implicitly defined by probability measures whose value is not directly computable. [44] have considered BNNs both as priors and variational family, where the ELBO gradients have been estimated using Stein Spectral Gradient Estimator [43]. However, due to its reliance on estimating intractable quantities from samples, this approach is not viable for computer vision tasks such as depth estimation, semantic segmentation or object classification with large number of classes, all of which display high-dimensional structure in both its inputs and outputs. In order to overcome this issue, we propose to first associate implicit priors with a Reproducing Kernel Hilbert Space (RKHS) and then defining a multi-output GP prior.

We consider $\mathcal{X} \subseteq \mathbb{R}^d$, where $d = CHW$ pertains to input images having C channels and $H \times W$ resolution, and $\mathcal{F} \subseteq \mathbb{R}^P$, where P is the output dimension depending on the task. For example, $P = HW$ for monocular depth estimation. Without loss of generality, we define $p(f(.))$ as a zero-mean multi-output stochastic process on $\mathcal{L}^2(\mathcal{F})$ whose index set is $\mathcal{X}$. Given two images x_i and x_j, $K(x_i, x_j) := \int f(x_i)^T f(x_j) dp(f(x_i), f(x_j))$ is the covariance function of the process, which is a $P \times P$ symmetric positive semi-definite matrix for each pair (x_i, x_j). We then posit a GP prior $\hat{p}(f(.))$ with zero mean and covariance function $K(.,.)$, and write its pair-wise joint distribution $\hat{p}(f(x_i), f(x_j))$ as follows:

$$\begin{pmatrix} f(x_i) \\ f(x_j) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} K(x_i,x_i) & K(x_i,x_j) \\ K(x_i,x_j) & K(x_j,x_j) \end{pmatrix}\right). \quad (7)$$

Writing the joint multivariate gaussian distribution for a batch of $B > 2$ images is straightforward: it is BP dimensional with zero mean vector, and its $BP \times BP$ covariance matrix contains B^2 blocks of $P \times P$ matrices, each of which is the evaluation of $K(.,.)$ at the corresponding pair of images. Matrices across the diagonal in the block describe the covariances between pixel locations for each image, whereas the off-diagonal ones describe the correlation between pixel locations of different images.

In the dense case, obtaining the inverse of the full covariance matrix is of complexity $O(B^3P^3)$ and carries a memory cost of $O(B^2P^2)$. Even if one is able to choose small B under the functional VI framework, this case would still be intractable for large P. A promising way of overcoming this would be to construct prior covariance functions with special structure across the P output dimensions. Recent work done in [11, 35, 48, 49] has highlighted that Bayesian CNNs do converge to Gaussian Processes as the number of channels of the hidden layers tends to infinity. In cases where activation functions such as relu and tanh are considered, and the architecture does not contain pooling layers, [11] shows that it is possible to exactly compute a covariance kernel which emulates the same behaviour as the Bayesian CNN, which is denoted as the *equivalent kernel*. In other words, given any Bayesian CNN of this form, in the limit of large number of channels, the function samples they generate come from a zero-mean Gaussian Process given by this covariance function (see [11] Figure 2 for an example). This covariance kernel can be computed very efficiently at cost which is proportional to a single forward pass through the equivalent CNN architecture with only one channel per layer, which is due to the fact that the resulting GP is independent and identically distributed over the output channels. Moreover, in the absence of pooling layers [35], the resulting kernel only contains the variance terms in its diagonal and all pixel-pixel covariances are 0. Thus, given a mini-batch of B input images, the corresponding prior kernel matrix $\boldsymbol{K}$ has only $O(B^2P)$ non-zero entries and can be written in block structure as follows:

$$\begin{pmatrix} K_{1,1} & \cdots & K_{B,1} \\ \vdots & \ddots & \vdots \\ K_{B,1} & \cdots & K_{B,B} \end{pmatrix} \quad (8)$$

Each sub-matrix $K_{i,j} = K(x_i, x_j)$ is diagonal, hence easy to invert and store. Let $\boldsymbol{K}_{:n,:n}$ denote the $nP \times nP$ sub-matrix obtained by indexing from the top-left corner of $\boldsymbol{K}$, where $n = 1, ..., B$, and consider the following block sub-matrix $\boldsymbol{K}_{:n+1,:n+1}$:

$$\begin{pmatrix} \boldsymbol{K}_{:n,:n} & \boldsymbol{K}_{:n,n+1} \\ \boldsymbol{K}^T_{:n,:n+1} & K_{n+1,n+1} \end{pmatrix} \tag{9}$$

Using the block-matrix inversion formula, we may write $\boldsymbol{K}^{-1}_{:n+1,:n+1}$ as follows:

$$\begin{pmatrix} \boldsymbol{A}_{:n,:n} & \boldsymbol{B}_{:n,n} \\ \boldsymbol{B}^T_{:n,n} & S^{-1}_{n,n} \end{pmatrix},$$
$$\boldsymbol{A}_{:n,:n} = \boldsymbol{K}^{-1}_{:n,:n}(\boldsymbol{I} + \boldsymbol{K}_{:n,n+1} S^{-1}_{n,n} \boldsymbol{K}^T_{:n,n+1} \boldsymbol{K}^{-1}_{:n,:n}),$$
$$\boldsymbol{B}_{:n,n} = \boldsymbol{K}^{-1}_{:n,:n} \boldsymbol{K}_{:n,n+1} S^{-1}_{n,n},$$
$$S_{n,n} = K_{n+1,n+1} - \boldsymbol{K}^T_{:n,:n+1} \boldsymbol{K}_{:n,:n} \boldsymbol{K}_{:n,n+1} \tag{10}$$

Where $S_{n,n}$ is the Schur-complement of $\boldsymbol{K}_{:n+1,n+1}$. This equivalence holds because $\boldsymbol{K}^{-1}_{:n+1,:n+1}$ is invertible if and only if $\boldsymbol{K}_{:n,:n}$ and $S_{n,n}$ are invertible. Starting from $n = 1$, $\boldsymbol{K}^{-1}_{:n+1,:n+1}$ can be recursively computed from $\boldsymbol{K}^{-1}_{:n,:n}$, so that we obtain $\boldsymbol{K}^{-1}$ in the last iteration. This algorithm is of complexity $O(B^2P)$, where B is much smaller than P since it is a batch-size, hence making functional VI applicable in the context of dense prediction tasks such as depth estimation and semantic segmentation. Additionally, the determinant of $\boldsymbol{K}$ may also be obtained efficiently by noting the following recurrence relation [38]:

$$\det(\boldsymbol{K}_{:n+1,:n+1}) = \det(\boldsymbol{K}_{:n,:n}) \det(S_{n,n}) \tag{11}$$

By efficiently and stably computing inverse covariance matrices with the same block structure as $\boldsymbol{K}$ and its respective determinants, we are able to replace $p(f^{\boldsymbol{X}})$ in (3) with the more convenient multi-output GP surrogate $\hat{p}(f^{\boldsymbol{X}})$. In this work we will only consider Bayesian CNN priors without pooling layers, which are most convenient in dense prediction tasks, in order to yield the structural advantages discussed above and leverage the methodology from [11, 35]. Nevertheless, given any square-integrable stochastic process, it is possible to estimate $K(x_i, x_j)$ using Monte Carlo (MC) sampling and then associating a GP prior with the estimated multi-output covariance function. This has been done in [35] in order to handle the cases where Bayesian CNN priors do contain pooling layers. Note that any cost involved in computing $\hat{p}(f^{\boldsymbol{X}})$ is only incurred during training.

Similarly, by choosing $q(f^{\boldsymbol{X}})$ to be a multi-output GP with mean function $h(.)$ and covariance function $\Sigma(.)$ parametrized by CNN mappings, we are able to compute the corresponding Gaussian KL divergence term in closed form. The expected log-likelihood term may be approximated with MC sampling, but in case of gaussian likelihood it can also be computed in closed form. For each pair

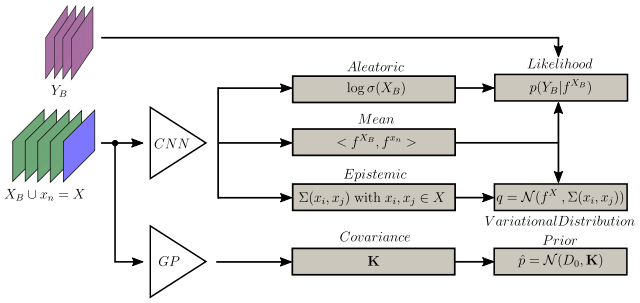

Figure 1. Overview of our functional VI approach. X_B is a batch of rgb inputs, x_n a newly generated one and D_0 is the mean function of the GP prior.

of inputs (x_i, x_j), we parametrize the covariance kernel as follows:

$$\Sigma(x_i, x_j) = \frac{1}{L} \sum_{k=1}^{L} g_k(x_i) \odot g_k(x_j) + D(x_i, x_j) \delta(x_i, x_j) \tag{12}$$

Where each $g_k(x_i), g_k(x_j)$ is a P dimensional feature mapping, $\odot$ denotes the element-wise product and $L < P$, so that the left-term is the diagonal part of a rank-L parameterization. For example, in depth estimation these can be obtained by defining $g(.)$ as a CNN having its output resolution associated with the P pixels and L output channels. $D(x_i, x_j)$ is a diagonal $P \times P$ matrix containing per-pixel variances which is considered only when $x_i = x_j$. This parametrization yields a $P \times P$ diagonal matrix for each pair of inputs, so that the full $BP \times BP$ covariance matrix has the same block structure as in (8). In this way $q(f^{\boldsymbol{X}})$ is able to account for posterior correlations between different images while being practical to train with mini-batches. Additionally, if one considers regression tasks whose likelihoods are of location-scale family, predictive variances can be computed in closed-form at no additional sampling cost (see supplementary material for an example under the berHu likelihood). In the case of discrete likelihoods, which includes semantic segmentation, computing entropy or mutual-information of the predictive distribution may also be done with a single forward pass plus a small number of gaussian samples, which adds negligible computational cost and is trivially paralellizable.

In practice, for each input image x, we may obtain all quantities of interest as an $R \times (LC + 3C)$ tensor by splitting the output channels of any suitable CNN architecture, where R is the desired output resolution, $C = 1$ for tasks such as monocular depth estimation or C equal to the number of classes for tasks such as semantic segmentation. In Figure 1 we display a more clear overview of the different components which form our proposed functional VI approach.

5. Related work

Monte Carlo Dropout (MCDropout) [9] interprets dropout as positing a variational family in weight-space and uses it at test time in order to compute epistemic uncertainty estimates. MCDropout has since then yielded applications in semantic segmentation tasks [19, 18, 20, 16, 33], moncular depth estimation [20], visual odometry [2] and active learning [10]. Despite being convenient to implement during training, the need for multiple forward passes at test time renders MCDropout impractical for both large network architectures (with many dropout layers) and tasks requiring high throughput, such as real-time computer vision. Alternatively, our proposed method allows one to obtain predictive epistemic uncertainty with a single forward pass and to consider a broad range of loss functions whose probabilistic counterparts are consistent with aleatoric uncertainty quantification.

In the ML literature, various approaches which consider the function space view of BNNs have been discussed in [13, 47, 31, 36, 22]. Gaussian Process Inference Networks (GPNet) [41] constitutes an alternative to inducing point methods on GPs, and shares some of the motivation of our work in that it also leverages the functional VI objective from [44] and chooses both variational family and prior to be GPs. In contrast to any of these, our work focuses on making training and inference practical in the context of dense prediction tasks, which is enabled by suitably parametrizing the variational GP approximation and exploiting special structure in the covariance matrices.

Recently [37] have proposed a scalable method which yields predictive epistemic uncertainty at the cost of a single forward pass. In contrast to it, ours naturally handles all forms of uncertainty, both at training and test times.

6. Results

In order to parametrize the variational GP approximation, we use the FCDenseNet 103 architecture [17] without dropout layers. We also adopt this architecture for all other baselines and experiments, using a dropout rate of 0.2. Even though our initial goal was to closely mimic the setup from [20], we were not able to reproduce their RMSprop results. Thus, in order to perform a clear comparison, we have decided to compare all methods with the exact same optimizer configurations. For MCDropout, we compute predictions using $S = 50$ forward passes at test time.

We choose $L = 20$ for the covariance parametrization in (12) and add a constant of 10^{-3} to its diagonal during training in order to ensure numerical stability. In order to implement the prior covariance kernel equivalent to a densely connected Bayesian CNN, which has been discussed in section 3, we use the PyTorch implementation made available by the authors in [11]. For both the segmentation and depth estimation experiments, we compute the equivalent kernel of a densely connected CNN architecture, composed of various convolutions and up-convolutions (see supplementary material), and add a white noise component of variance 0.1. For the depth experiments, we posit a prior mean of 0.5 while for segmentation we set it to 1.0. In order to generate the inducing inputs $\boldsymbol{X}'$ included in the KL divergence term from equation (3) during training, we randomly pick one image in the mini-batch and add per-pixel gaussian noise with variance 0.1.

6.1. Semantic Segmentation

In this section, we consider semantic segmentation on CamVid dataset [5]. All models have been trained with SGD optimizer, momentum of 0.9 and weight decay of 10^{-4} for 1000 epochs with batches of size 4 containing randomly cropped images of resolution 224×224, with an initial learning rate of 10^{-3} and annealing it every epoch by a factor of 0.998. Then we finish with training for one epoch on full-sized images with batch size of 1. We have considered this setup because, while performing our initial experiments by monitoring on the validation set, we have observed that our approach, even though it consistently benefits from fine-tuning on full-sized images in terms of its accuracy measures, the quality of its uncertainty estimates (in terms of calibration score [23]) has degraded significantly.

For our proposed method, we have used the Boltzmann likelihood with re-scaled logits as given in equation (4), which we denote as Ours-Boltzmann. Even though re-scaling logits provides no increase in flexibility to non-functional VI approaches, in order to have the same comparison setup, we chose to parametrize it in the same way for both the deterministic baseline and MCDropout: Deterministic-Boltzmann and MCDropout-Boltzmann, respectively.

From Table 1 we observe that our method performs best, both in terms of IoU score (averaged over all classes) and accuracy. In Figure 2 we display a test example of MCDropout-Boltzmann (top) and Ours-Boltzmann (bottom), where we have masked-out the void class label as yellow. We can see that the uncertainty estimates are reasonable, being higher on segmentation edges and unknown objects. We also include the calibration curve, as computed in [20], where the green dashed line corresponds to perfect calibration. In order to assess the overall quality of the uncertainty estimates, it is common to compute calibration plots for all pixels in the test set [20, 23]. Unfortunately, this is not feasible to compute for our functional VI approach, due to the fact that it captures correlations between multiple images, so that approximating the predictive distribution would require sampling from a high-dimensional non-diagonal gaussian. Thus, in order to enable a simple comparison which works for both Ours-Boltzmann and

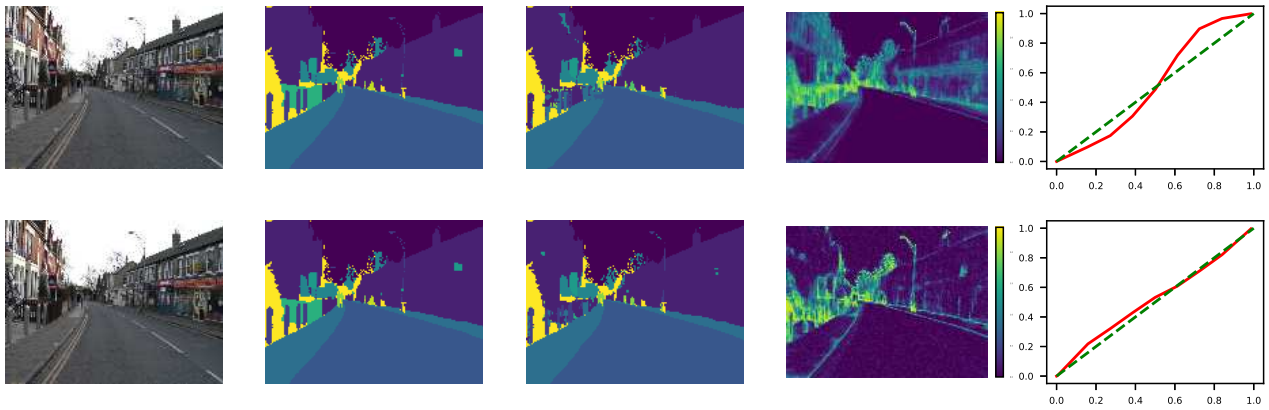

Figure 2. Semantic segmentation on CamVid. MCDropout-Boltzmann (top) and Ours-Boltzmann (bottom). From left to right: rgb input, ground truth, predicted, entropy, calibration plot (as depicted in [20])

MCDropout-Boltzmann, we compute the calibration score (see [23]) for each image in the test set and then average, which is given in Table 2.

Table 1. Results from training and testing on CamVid.

	IoU	Accuracy
Deterministic-Boltzmann	0.568	0.895
MCDropout-Boltzmann	0.556	0.893
Ours-Boltzmann	**0.623**	**0.905**

Table 2. Mean calibration score, computed with 10 equally spaced intervals, averaged over all test set examples. Lower is better.

	Mean Calibration
MCDropout-Boltzmann	0.058
Ours-Boltzmann	**0.053**

6.2. Pixel-wise Depth Regression

In this section, we consider depth estimation on Make3d dataset [40]. All models have been trained with AdamW optimizer [28] with constant learning rate and weight decay set to 10^{-4}. We have re-sized all images to a resolution of 168×224, and trained with a batch size of 4 for 3000 epochs. We consider our approach using 3 different likelihoods: Ours-Laplace, Ours-Gaussian and Ours-berHu (as derived in section 3.1.1). We compare with MCDropout-Laplace and two deterministic baselines: Deterministic-$\mathcal{L}_1$ and Deterministic-berHu using the reverse Huber loss [25].

Test results are displayed in Table 3, where MCDropout performs best on all accuracy metrics. To a certain extent, this happened because our proposed method is more sensitive to the choice of batch-size, due to the fact that the functional VI objective is not a lower bound to the log marginal likelihood of the dataset, so that it has underfitted slightly more than MCDropout-Laplace and deterministic methods. Additionally, we had to use a learning rate of 10^{-4}, as higher values would result in more unstable training for all our functional VI approaches.

In Figure 7 we plot one test prediction for MCDropout-Laplace (top) and Ours-Laplace (bottom). In this case, we observe one of the benefits of our approach: around the sky area in the image, MCDropout-Laplace is overconfident about its predicted depth map, while ours correctly outputs high predictive uncertainty. Note that this is not reflected in the calibration curves, as all pixels with depth greater than 70m are masked out due to long-range inaccuracies in the dataset [25].

In Table 4 we display the calibration scores for the probabilistic methods (see [23]), averaged over all test images, where Ours-Laplace performs slightly better than MCDropout-Laplace, despite not faring so well in terms of accuracy metrics.

Table 3. Results from training and testing on Make3d dataset.

	rel	log10	rms
Deterministic-$\mathcal{L}_1$	0.212	0.085	5.29
Deterministic-berHu	0.222	0.084	5.08
MCDropout-Laplace	**0.210**	**0.081**	**5.05**
Ours-Laplace	0.264	0.092	5.74
Ours-berHu	0.237	0.088	5.68
Ours-Gaussian	0.254	0.089	5.65

Table 4. Mean calibration score, computed with 10 equally spaced intervals, averaged over all test set examples. Lower is better.

	Mean Calibration
MCDropout-Laplace	0.427
Ours-Laplace	**0.409**
Ours-berHu	0.631
Ours-Gaussian	0.491

6.3. Inference time comparison

Let F be the inference time of one forward pass from a neural network on a RGB input. Our method's inference time (for obtaining predictive mean and uncertainty) is then $F + c_1$, while for MCDropout is $SF + c_2$, where c_1, c_2 are extra time costs needed to obtain the predictive uncertainties. In computer vision F is often the dominant term, since it often involves large network architectures, of which the FCDenseNet 103 architecture is an example. We have tested these claims by performing multiple runs on an NVIDIA RTX6000 GPU, the same device in which all models have been trained and tested. The inference times for depth estimation and semantic segmentation are displayed

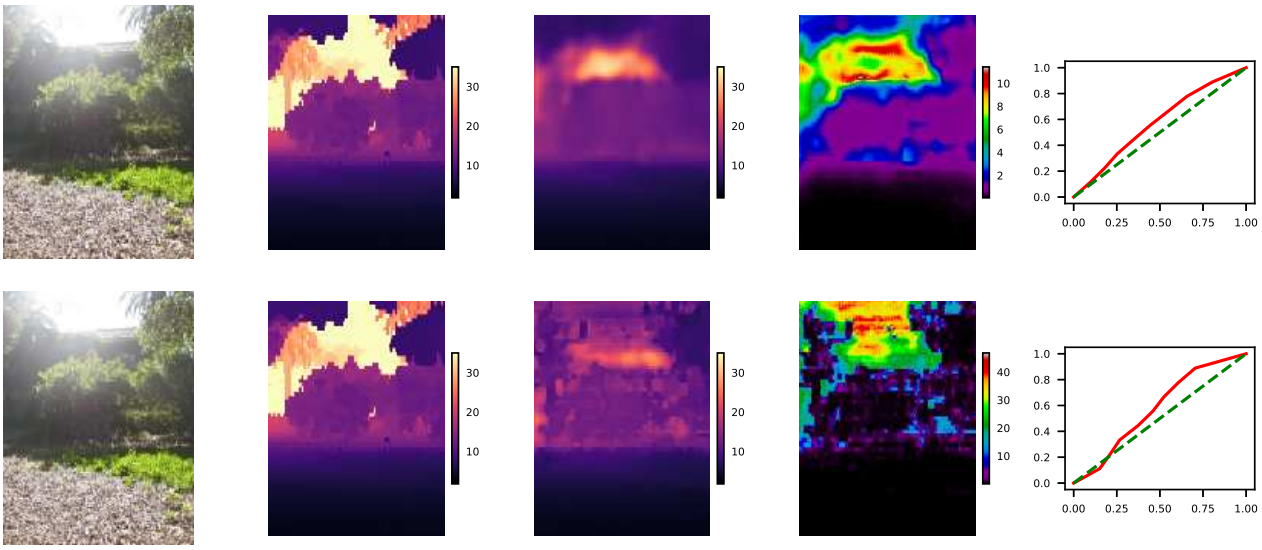

Figure 3. Depth estimation on Make3d. MCDropout-Laplace (top) and Ours-Laplace (bottom). From left to right: rgb input, ground truth, predictive mean, predictive standard deviation, calibration plot (as depicted in [23])

in Table 5 and Table 6, respectively. On depth estimation our method and deterministic had equivalent inference times. On segmentation c_1 depends on the number of gaussian samples taken, but is significantly cheaper than F and trivially amenable to parallelization, so that our method still displayed cost of same order as deterministic model. In both cases, MCDropout was approximately $S = 50$ times slower than its deterministic counterpart.

Table 5. Depth estimation on Make3D. Inference time comparison over 100 independent runs.

	mean ± std (ms)
Deterministic-$\mathcal{L}_1$	51.29 ± 1.88
Deterministic-berHu	51.28 ± 1.62
MCDropout-Laplace	2615.65 ± 13.75
Ours-Laplace	50.98 ± 1.74
Ours-berHu	51.43 ± 2.12
Ours-Gaussian	51.13 ± 2.20

Table 6. Semantic segmentation on CamVid. Inference time comparison over 100 independent runs.

	mean ± std (ms)
Deterministic-Boltzmann	111.64 ± 0.27
MCDropout-Boltzmann	5763.63 ± 1.95
Ours-Boltzmann	128.59 ± 1.86

7. Conclusion

We have proposed a method which, by leveraging the functional VI objective from [44], enables efficient training of Bayesian Deep Learning models and whose predictive inference requires only one forward pass, for any supervised learning task and network architecture. This is made possible by replacing the intractable BNN prior by a GP with covariance kernel as derived in [11], parametrizing the variational family as a GP with a suitably structured covariance kernel and by leveraging efficient algorithms for matrix inversion and determinant computation during training. Furthermore, we have discussed how to start with a well-defined loss function in regression and then derive its probabilistic counterpart in a way which is consistent with aleatoric uncertainty quantification, having provided the derivation of the berHu likelihood as an example.

Our framework may readily be applied to other pixel-wise supervised learning tasks. Extending to tasks which benefit from having pooling layers, such as object classification, is also possible but requires some caution. This is because Bayesian CNN priors which contain pooling layers no longer induce GPs which have the special covariance structure displayed in (8), given that pooling induces local correlations between different pixel locations [35].

As a direction of future work, it would be relevant to extend our proposed methodology to account for temporal information. This would be particularly important in monocular depth estimation, which is naturally prone to display high aleatoric uncertainty and would benefit from refined uncertainty estimates over consecutive time-frames [27]. Another direction of future work would be to overcome any potential underfitting occurring in pixel-wise regression tasks, as observed in our Make3D depth regression experiment, in which choosing more meaningful function priors and better variational distribution's covariance parametrizations could help.

Acknowledgements

Eduardo is supported by an EPSRC Industrial CASE scheme in collaboration with Arup. Paul is supported by EPSRC grant reference EP/P010040/1. We would like to thank Jan Czarnowski, Sajad Saeedi, Tristan Laidlow and all our reviewers for helpful insights and comments.

References

[1] Julian Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):192–225, 1974. 3

[2] Apratim Bhattacharyya, Mario Fritz, and Bernt Schiele. Long-term on-board prediction of people in traffic scenes under uncertainty. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1, 6

[3] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. Codeslam—learning a compact, optimisable representation for dense visual slam. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2560–2568, 2018.

[4] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015. 2

[5] Gabriel J Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009. 6

[6] Ronald Clark, Sen Wang, Andrew Markham, Niki Trigoni, and Hongkai Wen. Vidloc: A deep spatio-temporal model for 6-dof video-clip relocalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6856–6864, 2017. 1

[7] Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, and Niki Trigoni. Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017. 1

[8] Jan Czarnowski, Tristan Laidlow, Ronald Clark, and Andrew J Davison. Deepfactors: Real-time probabilistic dense monocular slam. *IEEE Robotics and Automation Letters*, 2020. 1

[9] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016. 6

[10] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1183–1192. JMLR. org, 2017. 1, 6

[11] Adrià Garriga-Alonso, Laurence Aitchison, and Carl Edward Rasmussen. Deep convolutional networks as shallow Gaussian processes. In *International Conference on Learning Representations*, 2019. 2, 4, 5, 6, 8

[12] Alex Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011. 1

[13] Danijar Hafner, Dustin Tran, Alex Irpan, Timothy Lillicrap, and James Davidson. Reliable uncertainty estimates in deep neural networks using noise contrastive priors. *arXiv preprint arXiv:1807.09289*, 2018. 1, 6

[14] James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013. 2

[15] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013. 1

[16] Po-Yu Huang, Wan-Ting Hsu, Chun-Yueh Chiu, Ting-Fan Wu, and Min Sun. Efficient uncertainty estimation for semantic segmentation in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 520–535, 2018. 1, 6

[17] Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 11–19, 2017. 6

[18] Michael Kampffmeyer, Arnt-Borre Salberg, and Robert Jenssen. Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1–9, 2016. 1, 6

[19] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015. 6

[20] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017. 1, 2, 3, 4, 6, 7

[21] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 4

[22] Mohammad Emtiyaz E Khan, Alexander Immer, Ehsan Abedi, and Maciej Korzepa. Approximate inference turns deep networks into gaussian processes.

In *Advances in Neural Information Processing Systems*, pages 3088–3098, 2019. 6

[23] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In *International Conference on Machine Learning*, pages 2801–2809, 2018. 6, 7, 8

[24] Tristan Laidlow, Jan Czarnowski, Andrea Nicastro, Ronald Clark, and Stefan Leutenegger. Towards the probabilistic fusion of learned priors into standard pipelines for 3d reconstruction. 2020. 1

[25] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth international conference on 3D vision (3DV)*, pages 239–248. IEEE, 2016. 2, 4, 7

[26] Sophie Lambert-Lacroix and Laurent Zwald. The adaptive berhu penalty in robust regression. *Journal of Nonparametric Statistics*, 28(3):487–514, 2016. 2, 4

[27] Chao Liu, Jinwei Gu, Kihwan Kim, Srinivasa G Narasimhan, and Jan Kautz. Neural rgb (r) d sensing: Depth and uncertainty from a video camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10986–10995, 2019. 8

[28] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 7

[29] Christos Louizos and Max Welling. Structured and efficient variational deep learning with matrix gaussian posteriors. In *International Conference on Machine Learning*, pages 1708–1716, 2016. 2

[30] Christos Louizos and Max Welling. Multiplicative normalizing flows for variational bayesian neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2218–2227. JMLR. org, 2017. 2

[31] Chao Ma, Yingzhen Li, and Jose Miguel Hernandez-Lobato. Variational implicit processes. In *International Conference on Machine Learning*, pages 4222–4233, 2019. 2, 6

[32] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992. 1

[33] Jishnu Mukhoti and Yarin Gal. Evaluating bayesian deep learning methods for semantic segmentation. *arXiv preprint arXiv:1811.12709*, 2018. 1, 6

[34] Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, Berlin, Heidelberg, 1996. 1

[35] Roman Novak, Lechao Xiao, Yasaman Bahri, Jaehoon Lee, Greg Yang, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations*, 2019. 4, 5, 8

[36] Tim Pearce, Mohamed Zaki, Alexandra Brintrup, and Andy Neely. Expressive priors in bayesian neural networks: Kernel combinations and periodic functions. *arXiv preprint arXiv:1905.06076*, 2019. 6

[37] Janis Postels, Francesco Ferroni, Huseyin Coskun, Nassir Navab, and Federico Tombari. Sampling-free epistemic uncertainty estimation using approximated variance propagation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2931–2940, 2019. 6

[38] Philip D Powell. Calculating determinants of block matrices. *arXiv preprint arXiv:1112.4379*, 2011. 5

[39] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003. 2

[40] Ashutosh Saxena, Min Sun, and Andrew Y Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):824–840, 2008. 7

[41] Jiaxin Shi, Mohammad Emtiyaz Khan, and Jun Zhu. Scalable training of inference networks for gaussian-process models. In *International Conference on Machine Learning*, pages 5758–5768, 2019. 2, 3, 6

[42] Jiaxin Shi, Shengyang Sun, and Jun Zhu. Kernel implicit variational inference. In *International Conference on Learning Representations*, 2018. 2

[43] Jiaxin Shi, Shengyang Sun, and Jun Zhu. A spectral approach to gradient estimation for implicit distributions. In *International Conference on Machine Learning*, pages 4651–4660, 2018. 4

[44] Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional Variational Bayesian Neural Networks. In *International Conference on Learning Representations*, 2019. 2, 3, 4, 6, 8

[45] Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574, 2009. 2

[46] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks. *The International Journal of Robotics Research*, 37(4-5):513–542, 2018. 1

[47] Ziyu Wang, Tongzheng Ren, Jun Zhu, and Bo Zhang. Function space particle optimization for bayesian neu-

ral networks. In *International Conference on Learning Representations*, 2019. 6

[48] Greg 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. 4

[49] Greg Yang. Wide feedforward or recurrent neural networks of any architecture are gaussian processes. In *Advances in Neural Information Processing Systems*, pages 9947–9960, 2019. 4