Trust your robots! Predictive uncertainty estimation of Neural Networks with scalable Gaussian Processes

Anonymous Author(s)
Affiliation
Address
email

Abstract: This paper presents a probabilistic framework to obtain both reliable and fast uncertainty estimates for predictions with Deep Neural Networks (DNNs). Our main contribution is a practical and principled combination of DNNs with scalable Gaussian Processes (GPs). We prove theoretically that DNNs can be seen as a special case of scalable GPs, namely mixtures of GP experts (MoE-GP), and we devise a learning algorithm that brings the derived theory into practice. In experiments from two different robotic tasks – inverse dynamics of a manipulator and object detection on a micro-aerial vehicle (MAV) – we show the effectiveness of our approach in terms of predictive uncertainty, scalability to large dataset, and run-time efficiency on a Jetson TX2. We thus argue that our approach can pave the way towards reliable and fast robot learning systems with uncertainty awareness.

Keywords: Robotic Introspection, Bayesian Deep Learning, Gaussian Processes

1 Introduction

This work aims to provide an algorithm that can estimate uncertainty of DNN predictions reliably and fast, and at the same time, is suited for integration into a large range of robotic systems. Generic solutions to this problem are crucial for safe robot learning and introspection [1, 2], giving the robots with an ability to assess their own failure probabilities and to alter their behaviors towards safety. While the state of the art is advancing [3, 4, 5], we still face practical difficulties in two main domains. First, the current methods are often not efficient at test-time, e.g. for a single input, the methods require multiple predictions from several copies of a model [4] or samples from the model's distribution [3]. And second, in general these methods do not provide reliable uncertainty estimates when compared to GPs - often known as the golden standard of probabilistic machine learning [6].

Therefore, we propose to estimate the *predictive uncertainty* of DNNs using MoE-GPs [7, 8] - a scalable variant of GPs that divides the input space into smaller local regions using a *gating function*, where individual GPs called *experts* learn and make predictions (see figure 1). To do so, we provide both theoretic foundations and a practical learning algorithm. First, we formally derive a connection between DNNs and MoE-GPs. As a result, we show how MoE-GPs with a DNN-based kernel [9] can estimate the predictive uncertainty of DNNs. Moreover, we devise a learning algorithm that brings the derived theory into practice. Our solution involves a gating function that strictly divides the input data into smaller subsets by performing clustering in kernel space, and we propose the concept of a patchwork prior, mitigating the problem of discontinuity between local GP experts at their boundaries. For efficiency, we further propose to exploit active learning and model compression techniques.

Our approach has several favorable features for many classification and regressions tasks in robotics. At run-time, it maintains the predictive power of a DNN, and its uncertainty estimates do not require combining multiple predictions of DNNs. Moreover, we inherit the benefits of MoE-GPs for uncertainty estimates. These include an improved scalability when compared to a GP with the neural tangent kernel [9] (NTK), as well as natural support for distributed training. We contend that such probabilistic methods must run efficiently onboard a robot and scale to large datasets. Therefore, we not only provide ablation studies and evaluations against the state-of-the-art (SOTA), but we also show that our method (i) scales to approximately 2 million data points for the task of learning inverse

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

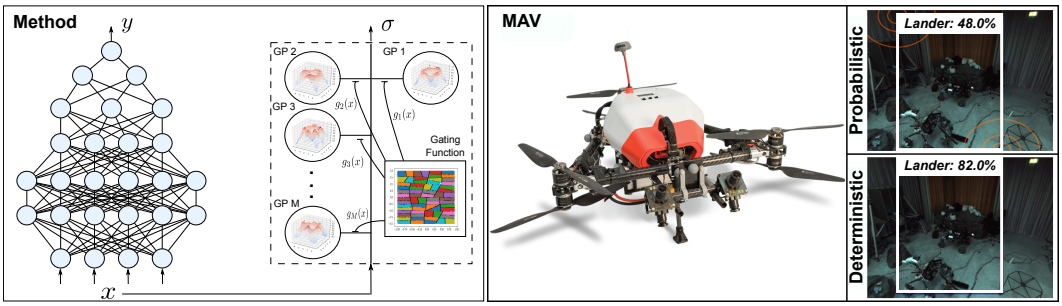

Figure 1: Our method (left) computes uncertainty σ of Neural Network predictions $\mathbf{y}$ using MoE-GPs with the Neural Tangent Kernels (NTK). With this, a MAV (right) performs probabilistic object detection at an interactive frame rate, reducing overconfidence of an object detector. Only a feedforward network is shown but our method also applies to convolution and recurrent DNNs.

dynamics, and (ii) on a Jetson TX2 of a MAV, runs more than 12 times faster than the widely used MC-dropout [3], while performing object detection within a scenario for planetary exploration.

Contributions In summary, our main contribution is a novel method for estimating predictive uncertainty of DNNs with scalable GPs (section 3.1), backed up by (i) a theoretical connection between DNNs and MoE-GPs (section 3.2), (ii) a learning algorithm for its applicability in practice (section 3.3), and (iii) an exhaustive empirical evaluation that demonstrates the effectiveness of our approach (section 4). Importantly, the accompanying code provides an efficient implementation of the proposed ideas in PyTorch, and the video provides qualitative results. To our knowledge, we are the first to show that you can use GPs to estimate the uncertainty of DNNs on a Jetson TX2 of a MAV.

2 Related Work

Many researchers explored the idea of robotic introspection [2], i.e. robots that reason about *"when they don't know"* in order to avoid catastrophic failures. So far, many proposed methods used the robots' past experiences of failures [10, 11, 12], or detect the inconsistencies within the predictors [13, 14]. These works provide strong evidence that introspection improves the reliability of the robots. Other methods are based on uncertainty estimation [1], where learning algorithms such as DNNs use probability theory to express their own reliability [15, 16, 17, 18, 19]. Commonly, these works use a technique called MC-dropout [3]. However, the intense computational burden in estimating uncertainty via sampling or ensembles [3, 4] limits their applicability in real-time systems.

Several techniques without sampling or ensembles have been presented for run-time efficiency. These methods often use either Dirichlet distributions or functional Bayesian Neural Networks [20, 21]. Unfortunately, the applicability of Dirichlet distributions is limited to classification, while functional Bayesian Neural Networks face hurdles in scaling to large datasets without approximations [21]. We recommend the recent works of He et al. [22] and Adlam et al. [23], who use NTK-based GPs for uncertainty estimation of DNNs and provide useful insights. However, we focus on a technique that can be readily deployed on a real-time system such as a MAV. Hence, we take a different path to ensembles of infinite width DNNs [22, 23]. We find uncertainty propagation techniques [24, 25, 26] to be practical as these methods are training-free, i.e. assume existing and already well-trained DNNs. Recently, there are also other important methods without sampling or ensembles [27, 28]. However, they assume substantial modifications to the original DNNs, which limit their applicability in practice.

In addition, our theoretic contribution (section 3.2) extends on how DNNs are related to GPs (pioneered by Neal [29] and others [30, 9, 31]), where we derive a relationship between DNNs and MoE-GPs as opposed to a single GP. Theoretic insights that connect DNNs and scalable GPs may pave the way towards their application, as GPs alone do not scale to big data required by DNNs. Lastly, we extend previous works on MoE-GPs (section 3.3). MoE-GPs are efficient but suffer from non-smooth uncertainty estimates [32]. Park and Apley [33] show an elegant solution called patchwork kriging. By augmenting data at the boundaries between GP experts, the experts are forced to produce similar predictions at their boundaries. Yet, as patchwork kriging is limited to low-dimensional inputs [33], we propose several techniques that extend it to higher dimensions, like images.

3 Materials and Methods

We describe a method that addresses the problem of quantifying uncertainty estimates in DNN predictions, without sampling or ensembles. First, we introduce the main concept of estimating predictive uncertainty with scalable GPs, followed by theoretic results and our learning algorithm.

3.1 Main Idea: Fast Uncertainty Estimation with Gaussian Processes

Figure 1 visualizes our approach. The proposed predictive model uses an existing, and already well-trained DNN for accurate predictions. At the same time, the method offers the possibility to obtain the predictors' reliable uncertainty in a closed form solution, using a DNN-based MoE-GPs. Intuitively, we can linearize and transform DNNs around a mode to obtain a linear model. As any linear models are GP regressors from a function space view, we can get a GP-based representation of DNNs with a DNN-based kernel, the NTK. We describe below how the obtained GPs can estimate the uncertainty of DNNs, but cannot replace the DNN predictions. Moreover, as full GPs do not scale to big data, we choose to use MoE-GPs, which divides the data into smaller subsets and form many smaller GPs per these subsets. This results in the proposed combination of DNNs and scalable GPs.

Fundamentals First, we introduce our notation. Considering a supervised learning task on input-output pairs $\mathcal{D} = \left\{\mathcal{X}, \mathcal{Y}\right\} = \left\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\right\}_{i=1}^{N}$, where $\boldsymbol{x}_i \in \mathbb{R}^D$, $\boldsymbol{y}_i \in \mathbb{R}^K$, we describe a DNN as a parametrized function $f_{\boldsymbol{\theta}} : \mathbb{R}^D \to \mathbb{R}^K$, where $\boldsymbol{\theta} \in \mathbb{R}^P$. Here, learning typically seeks to obtain an empirical risk minimizer of the loss function, i.e. $\min_{\boldsymbol{\theta}} \quad \frac{1}{|\mathcal{D}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y}) + \frac{\delta}{2}\boldsymbol{\theta}^T\boldsymbol{\theta}$ where δ is an L_2-regularizer, and mini-batches $\mathcal{B} \subset \mathcal{D}$ are used to find a local maximum-a-posteriori (MAP) solution $\hat{\boldsymbol{\theta}}$. We assume a twice differentiable and strictly convex loss function $\mathcal{L}$, e.g. mean squared error (MSE), and piece-wise linear activations in $f_{\boldsymbol{\theta}}$ (e.g. RELU). For a clear exposition, we drop the indices i.

To set the scene for the paper, we define the Neural Linear Models (NLMs), which has an ability to estimate a DNN's predictive uncertainty for many loss functions [24, 26, 34, 35, 31]. To do so, consider a transformed dataset $\widetilde{\mathcal{D}} = \left\{\mathcal{X}, \widetilde{\mathcal{Y}}\right\}$ with the pseudo-output $\widetilde{\boldsymbol{y}} := \boldsymbol{J}_f(\boldsymbol{x})\hat{\boldsymbol{\theta}} - \boldsymbol{H}_{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{y})^{-1}\boldsymbol{R}_{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{y})$. This transformation uses the model derivatives of the underlying DNN around $\hat{\boldsymbol{\theta}}$, namely the Jacobian $\boldsymbol{J}_f(\boldsymbol{x}) := \partial f_{\boldsymbol{\theta}}(\boldsymbol{x})/\partial\boldsymbol{\theta}^T \in \mathbb{R}^{K\times P}$, the Hessian $\boldsymbol{H}_{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{y}) := \partial^2\mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y})/\partial f_{\boldsymbol{\theta}}(\boldsymbol{x})^T\partial f_{\boldsymbol{\theta}}(\boldsymbol{x}) \in \mathbb{R}^{K\times K}$ and the residuals $\boldsymbol{R}_{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{y}) := \partial\mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y})/\partial f_{\boldsymbol{\theta}}(\boldsymbol{x}) \in \mathbb{R}^K$. With that, the NLMs are written as,

$$\widetilde{\boldsymbol{y}} = \boldsymbol{J}_f(\boldsymbol{x})\boldsymbol{\theta} + \epsilon \quad \text{with} \quad \epsilon \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{H}_{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{y})^{-1}) \quad \text{and} \quad p(\boldsymbol{\theta}) \sim \mathcal{N}(\boldsymbol{0}, \delta^{-1}\boldsymbol{I}). \tag{1}$$

For commonly used loss functions such as cross entropy and MSE, the covariance $\boldsymbol{\Sigma}$ of DNN predictions $f_{\boldsymbol{\theta}}$ is equal to that of the NLMs' $\widetilde{\boldsymbol{y}}$ [24, 31]. This means that for the same input $\boldsymbol{x}$, the NLMs can estimate the predictive uncertainty of DNNs. Yet, $\widetilde{\boldsymbol{y}}$ cannot be used to replace $f_{\boldsymbol{\theta}}$, due to their differences in $f_{\boldsymbol{\theta}}$ and $\widetilde{\boldsymbol{y}}$. In the Appendix, we provide concrete examples and their derivations.

Main Idea Let's now apply a kernel trick [36] to the NLMs, so that we can obtain an effective combination of DNNs with GPs. To explain, we can further map the NLM to a function space, as opposed to working in the weight space. Doing so, a well known function space formulation of linear models [36] turns the NLM into a GP with the kernel $\boldsymbol{K} = \frac{1}{\delta}\boldsymbol{J}_f(\boldsymbol{x})^T\boldsymbol{J}_f(\boldsymbol{x})$, which is also known as the *Network Tangent Kernel* (NTK). Thus, GPs with NTK can be used to estimate the predictive uncertainty of DNNs, as their equivalent NLMs can. Here, what motivates the formulation is the idea of deep kernel machines [37], i.e. we learn the kernel representations as oppose to hand designing the features and the kernels, and in our case, the kernel representations are the tangents of DNNs' Jacobians. Importantly, doing so, we get a training-free combination of DNNs with GPs, as we keep DNN predictions the same as the MAP, while estimating their uncertainty using GPs with NTK.

Unfortunately, such combinations of DNNs with GPs can only work, if we resort to scalable GPs. This is because the computational costs of full GPs are intractable $\mathcal{O}(N^3)$, and scalable GPs aim to find a computationally tractable $\mathcal{O}(\bar{N}^3)$ such that $\bar{N} << N$. While we leave the theoretic motivation to section 3.2, we use MoE-GPs with NTK, in order to scale GPs to big data that DNNs assume. A MoE-GP consists of M experts of GPs or learners $\mathcal{F} = \left\{\widetilde{f}_{\mathrm{GP}_1}, \cdots, \widetilde{f}_{\mathrm{GP}_M}\right\}$, and a gating function $g : \mathbb{R}^D \to \Delta^{M-1}$ that maps any input $\boldsymbol{x}$ to $g(\boldsymbol{x}) = [g_1(\boldsymbol{x}), \cdots, g_M(\boldsymbol{x})]$ [7, 8]. Each expert of the MoE learns and predicts within a subset of the input domain, and a gating function generates these subsets.

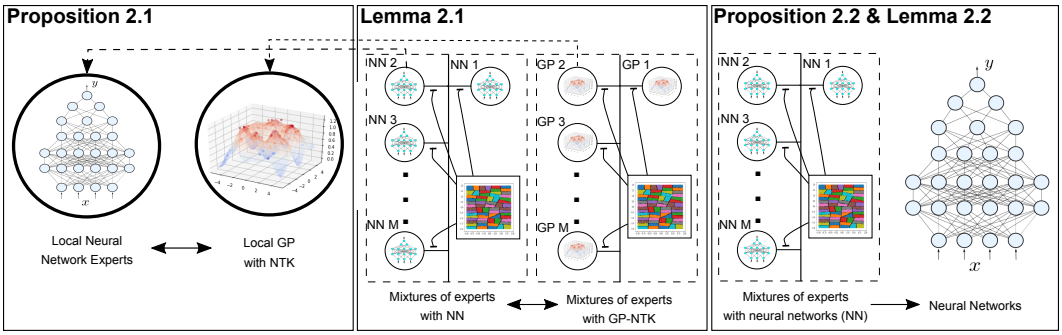

Figure 2: We visualize main results. Detailed theorems and their proofs are in the Appendix.

Specifically, we choose a gating function $g_m(\boldsymbol{x}) = 1$ in just one coordinate for each input [38]. The subscripts $m = 1, 2, ..., M$ denote the mth expert throughout the paper. Then, we write a MoE-GP as,

$$\widetilde{\boldsymbol{y}} = \sum_{m=1}^{M} g_m(\boldsymbol{x}) \widetilde{f}_{\text{GP}_m}(\boldsymbol{x}) + \epsilon \quad \text{with} \quad \widetilde{f}_{\text{GP}_m}(\boldsymbol{x}) \sim \text{GP}(\boldsymbol{0}, \frac{1}{\delta_m} \boldsymbol{J}_{f_m}(\boldsymbol{x})^T \boldsymbol{J}_{f_m}(\boldsymbol{x})). \tag{2}$$

As depicted in figure 1, the gating function $g_m(\boldsymbol{x})$ assigns the input data $\boldsymbol{x}$ to the mth subset, and individual GPs $\widetilde{f}_{\text{GP}_m}(\boldsymbol{x})$ learn and make predictions for the assigned data within the mth subset. Consequently, the generative model and the predictions $\mathcal{N}(\widetilde{\boldsymbol{y}}_m^*, \boldsymbol{\Sigma}_m(\boldsymbol{x}^*))$ on the new test datum $\boldsymbol{x}^*$ are:

$$\begin{bmatrix} \widetilde{\boldsymbol{y}}_m \\ \widetilde{f}_{\text{GP}_m} \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K}_m + \sigma_{0,m}\boldsymbol{I} & \boldsymbol{k}_{m,*} \\ \boldsymbol{k}_{m,*} & \boldsymbol{k}_{m,**} \end{bmatrix}\right) \quad \text{and} \quad \begin{aligned} \widetilde{\boldsymbol{y}}_m^* &= \boldsymbol{k}_{m,*}^T (\boldsymbol{K}_m + \sigma_{0,m}\boldsymbol{I})^{-1} \widetilde{\boldsymbol{y}}_m, \\ \boldsymbol{\Sigma}_m &= \boldsymbol{k}_{m,**} - \boldsymbol{k}_T^{m,*} (\boldsymbol{K}_m + \sigma_{0,m}\boldsymbol{I})^{-1} \boldsymbol{k}_{m,*} + \sigma_{0,m}, \end{aligned}$$

where, $\boldsymbol{K}_m = \boldsymbol{K}_m(\mathcal{X}, \mathcal{X})$, $\boldsymbol{k}_{m,*} = \boldsymbol{k}_m(\mathcal{X}, \boldsymbol{x}^*)$ and $\boldsymbol{k}_{m,**} = \boldsymbol{k}_m(\boldsymbol{x}^*, \boldsymbol{x}^*)$. Also, data uncertainty is captured by $\sigma_{0,m}\boldsymbol{I}$. Finally, we stress the two points. First, as shown above, the generative model of a GP is defined per subset, and therefore, the GP experts are smaller than the one on the entire dataset. This reduces the computational complexity of GPs. Second, the covariance computations are in a closed form. Thus, we neither need sampling nor ensemble [3, 4]. Using so-called Lanczos approximation [39], which approximates the kernel matrix inversion with matrix multiplications (suited for GPUs), we can compute the uncertainty estimates from GPs, with a constant time complexity [39].

For classification, we use GP regression for confidence calibration [5, 40]. For a class c:

$$p(c|\boldsymbol{z}_m) = \text{softmax}\left(\frac{\boldsymbol{z}_m}{T_m}\right) \text{ with } T_m = \sqrt{1 + \lambda_{m,0} \boldsymbol{\Sigma}_m(\boldsymbol{x}^*)}, \tag{3}$$

where $\boldsymbol{z}_m$ is the activation, T_m is the temperature scaling factor, which is a function of a scaling constant $\lambda_{m,0}$, and GP regression uncertainty $\boldsymbol{\Sigma}_m(\boldsymbol{x}^*)$. In this way, high prediction uncertainty will reduce the corresponding $p(c|\boldsymbol{z}_m)$ by increasing the T_m, thereby calibrating the confidences [5, 41, 40].

3.2 On Theory: Neural Networks as Scalable Gaussian Processes

So far, we have outlined our main idea - using the function space view of NLMs and dividing the input space into smaller subsets, we can form smaller GPs per division. These GPs then estimate the predictive uncertainty of DNNs. Now, we discuss the theoretical foundations on how DNNs can be cast as MoE-GPs with NTK. Due to space constraints, we refer the reader to the Appendix for in depth treatment, where we provide various background materials, our theorems and their proofs. In section 3.3 we explain our learning algorithm while here, we briefly summarize the main results.

Main result Figure 2 summarizes our findings. Consider a mixtures of experts model, where the experts are DNNs and a gating function strictly divides the input space, i.e only one local DNN expert per division (MoE-DNN). Then, instead of the MAP parameter estimates $\hat{\boldsymbol{\theta}}$, we consider Bayesian DNNs with Gaussian distributions $p(\boldsymbol{\theta}|\mathcal{D})$, i.e representing DNNs as probability distributions over

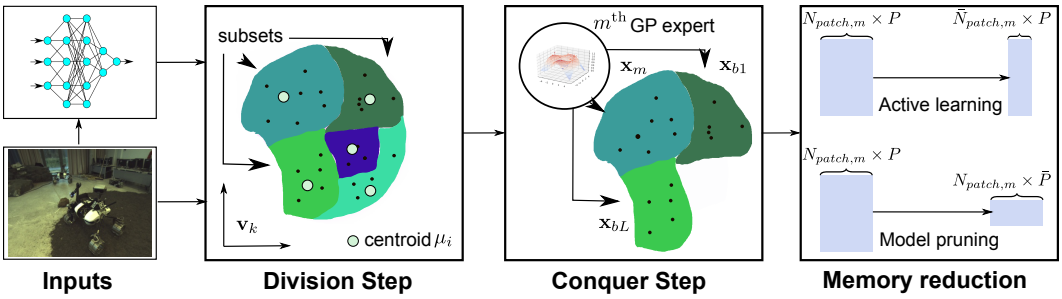

Figure 3: The proposed algorithm involves (i) the division of input data using Jacobians of already trained DNNs, (ii) training of GP experts within their partitions and neighboring regimes, and (iii) combining active learning and model pruning techniques to reduce space complexity.

their parameters (obtained using [42, 43, 35]). As a first step, as a specific instance of Khan et al. [31], we show that the DNN experts have mathematically equivalent Gaussian distributions with GPs using NTK. We further show that MoE-DNNs and MoE-GPs have equivalent Gaussian distributions, establishing the duality of the two models in a Bayesian sense. Main insight to the former is the probabilistic independence between the local experts due to a strict division of the input space.

Next, we establish a connection between a DNN, and a MoE-GP as shown in figure 2, in order to increase the applicability of our theoretic results. To do so, we point out that the input-prediction relationships of a single DNN and a MoE-DNN are equivalent, if all the local DNN experts are the same as the single DNN. Under the given conditions, we derive that a single DNN can be cast as a MoE-GP, with an assumption that the data is stationary. The resulting step yields the approximation error of $\|\boldsymbol{K}(\mathcal{X},\mathcal{X}) - \boldsymbol{K}_{true}(\mathcal{X},\mathcal{X})\|_{\mathrm{F}}^2 = \sum_{ij} \boldsymbol{K}(\boldsymbol{x}_i,\boldsymbol{x}_j)^2 - \sum_{m=1}^{M} \sum_{ij\in\mathfrak{V}_m} \boldsymbol{K}(\boldsymbol{x}_i,\boldsymbol{x}_j)^2$ where $\boldsymbol{K}_1 = \boldsymbol{K}_2 = \cdots = \boldsymbol{K}_M$ for all M GP experts (while it is still possible to keep different hyper-parameters δ_m and σ_m), and $\boldsymbol{K}_{true}$ is the kernel matrix of the true DNN-equivalent GP with NTK. This means that less approximation error occurs when less correlated data points are assumed to be independent by MoE-GP, while strongly correlated points by NTK are within the same GP experts.

3.3 From Theory to Practice: A Practical Learning Algorithm

Now, we attempt to bring our theoretical results into a practical tool for robotics. As shown in figure 3 our solution involves the division of data into several partitions. Then, in a conquer step, we train GP experts within the partitions. Moreover, we propose concepts to reduce the memory complexity.

Division Step We design a gating function that divides the input data into smaller regimes s.t. highly correlated data points are grouped together whilst the points with weak correlations are separated apart. To do so, we perform NTK PCA, i.e. kernel PCA that uses NTK to reduce the dimensions of the data. Then, K-means is applied to form the clusters. Concretely, NTK PCA reduces the input data $\boldsymbol{x}_i$ for $i = 1,...,N$ into the low dimensional principal components $\boldsymbol{v}_k = \sum_{i=1}^{N} \boldsymbol{\alpha}_{ik} \boldsymbol{J}_f(\boldsymbol{x})^T \boldsymbol{J}_f(\boldsymbol{x}_i)$, where $\boldsymbol{\alpha}$ is the eigen-vector of $\widetilde{\boldsymbol{K}}\boldsymbol{\alpha}_k = \lambda_k \boldsymbol{\alpha}_k$ with a centered kernel matrix $\widetilde{\boldsymbol{K}} = \boldsymbol{K} - 2\mathbf{1}_N \boldsymbol{K} + \mathbf{1}_N \boldsymbol{K} \mathbf{1}_N$ for the matrix of ones $\mathbf{1}_N \in \mathbb{R}^{N\times N}$. Then, the centroid μ_i and division labels $\boldsymbol{c}$ are computed, iterating:

$$\boldsymbol{c}_i = \arg\min \sum_{i}^{M} \sum_{\boldsymbol{v}} \|\boldsymbol{v}_i - \mu_i\|^2 \quad \text{and} \quad \mu_i = \frac{\sum_i^M \mathbf{1}\{\boldsymbol{c}_i = i\} \boldsymbol{v}_i}{\sum_i^M \mathbf{1}\{\boldsymbol{c}_i = i\}}. \tag{4}$$

The proposed technique gives a solution for Kernel K-means [44], which minimizes the error $\|\boldsymbol{K}(\mathcal{X},\mathcal{X}) - \boldsymbol{K}_{true}(\mathcal{X},\mathcal{X})\|_{\mathrm{F}}^2$ with a balancing normalization [45]. As kernel PCA does not scale to large datasets, we use 2-step kernel K-means [46], which uses randomly sampled and less data to compute the cluster centroids. We refer to Chitta et al. [46] for details.

Conquer Step Next, we form the individual GP experts per divided local regime. A naive strategy is to form a single GP per regime. Unfortunately, such a model suffers from discontinuity in predictions

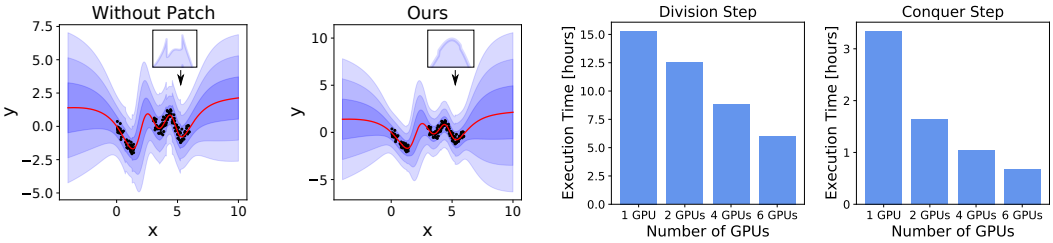

Figure 4: (Left) We visualize predictive uncertainty of DNNs with and without the proposed patchwork prior. The black dots are train data points, and the red line shows the predictions of a deterministic network. Blue shades show up to three standard deviations. Without the patch prior, sharp peaks are observed in uncertainty estimates. (Right) Training time for approximately 2 million data points is shown for a varying the number of GPUs. We learned 512 GP experts per joint torques.

at the boundaries [32]. For example, input points at the cluster boundaries may yield different predictions from the surrounding GPs, causing sharp peaks in predictions. Therefore, we propose

$$\boldsymbol{K}_{m,\text{patch}} = \left[\boldsymbol{J}_{f_m}(\boldsymbol{x}_m), \boldsymbol{J}_{f_m b1}(\boldsymbol{x}_{b1}), \cdots \boldsymbol{J}_{f_m bL}(\boldsymbol{x}_{bL})\right]^T \left[\boldsymbol{J}_{f_m}(\boldsymbol{x}_m), \boldsymbol{J}_{f_m b1}(\boldsymbol{x}_{b1}), \cdots \boldsymbol{J}_{f_m bL}(\boldsymbol{x}_{bL})\right] \quad , \tag{5}$$

for the mth GP expert. We include the neighboring GP experts $\boldsymbol{J}_{f_m b1}, ..., \boldsymbol{J}_{f_m bL}$ into the prior of the mth expert. This includes the information of neighboring GPs into the prior of the mth GP expert. Intuitively, the discontinuities occur at the shared boundary regimes of two GP experts. Our GP prior removes such boundary regimes locally by taking into account the neighboring regimes.

Active Uncertainty Learning The complexity of each GP expert is bounded to $O(N^3_{patch,m})$, where the number of data points $N_{patch,m}$ includes the data points of neighboring GPs: $N_{patch,m} = N_m + N_{b1} + \cdots + N_{bk}$, resulting in added costs $N_m < N_{patch,m}$. To mitigate, we perform active learning on the neighboring GP experts, and thus choose fewer, but the most informative points [47, 48]. Intuitively, as the neighboring data are only included for reducing the discontinuity problem, we may select fewer data points. Concretely, we use the following steps: (i) for the neighboring GPs of the mth expert, we draw an initial subset and train the GPs. (ii) Using the trained GP models, we rank the remaining data (stored as a pool) by uncertainty (queries generated by uncertainty sampling). (iii) the GP expert is then updated with the most uncertain point. These steps repeat until N_{bk} is reduced to a desired, smaller value. In this way, while forming a patchwork prior, the neighboring GPs actively choose the neighboring data they want to learn from. As a result, we obtain: $\bar{N}_{patch,m} < N_{patch,m}$.

Gaussian Process Compression We further reduce the complexity of the algorithm by exploiting model compression techniques. Note that the Jacobians $\boldsymbol{J}_f(\boldsymbol{x})$ are $N_{patch,m}$ by P matrices where P is the total number of parameters in DNNs. As $\boldsymbol{J}_f(\boldsymbol{x})$ represents the sensitivity of each parameter to the output, the elements of $\boldsymbol{J}_f(\boldsymbol{x})$ that belong to an unimportant DNN parameter can be removed. To do so, we propose a two-staged compression technique: (i) rank the DNN parameters by their importance using existing pruning methods and remove the corresponding elements of $\boldsymbol{J}_f(\boldsymbol{x})$ for all m. (ii) Per GP expert, rank the elements of $\boldsymbol{J}_{f_m}(\boldsymbol{x})$ by a metric $|\sum \boldsymbol{J}_{f_m}(\boldsymbol{x})|$, as smaller absolute values contribute less to the kernel. The first step is targeted at removing redundant parameters in DNNs (for the division step) while the second step is targeted at each individual expert, resulting in $\bar{P} < P$.

4 Experiments and Evaluations

We provide 5 sets of experiments to not only validate the method, but also to show the benefits of the proposed formulation namely performance, scalability and run-time. Implementation details can be found in the Appendix, and importantly, the accompanying video shows the experiments with a MAV.

Toy illustration With a toy regression on the Snelson dataset, we show (i) on how MoE-GPs can estimate the predictive uncertainty of DNNs, and (ii) the ability of the patchwork prior to produce smoother uncertainty estimates. For this, we train a single hidden layer Multi-layer perceptron (MLP) with 200 units and a tanh activation. Evaluating for out-of-distribution (OOD) data (points

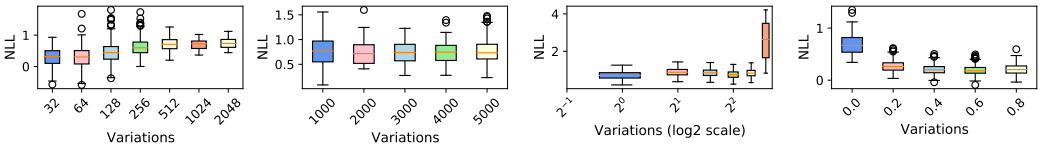

Figure 5: The effects of each hyperparameter is shown with normalized test NLL, by varying them in different steps (Variations), and fixing the others to default settings. Lower the better.

Table 1: The results of inverse dynamic experiments are reported using NLL. Lower the better.

Train	Test	**MC-dropout**	**rMC-dropout**	**Laplace**	**rLaplace**	**Ours**
sarcos	sarcos	1.398±0.012	1.398±0.012	1.392±0.014	1.392±0.014	**1.035±0.039**
sarcos	kuka1	**16.21±0.866**	20.58±1.144	25.88±1.354	24.55±2.746	21.53±2.268
sarcos	kuka2	**15.63±0.858**	20.17±0.988	25.59±1.079	24.42±2.257	20.92±2.236
sarcos	kukasim	**51.09±11.14**	61.40±12.921	77.13±15.40	74.03±13.23	68.95±4.003
kuka1	kuka1	1.611±0.007	1.620±0.006	1.676±0.013	1.719±0.071	**1.347±0.013**
kuka1	kuka2	1.349±0.019	1.33±0.006	**1.310±0.005**	**1.310±0.008**	1.315±0.003
kuka1	kukasim	42.14±2.869	48.76±0.566	60.37±0.799	59.74±1.571	**1.348±0.012**
kuka1	sarcos	22.24±4.288	42.36±2.398	122.68±13.32	1837±2431.3	**1.356±0.014**
kuka2	kuka2	1.443±0.005	1.423±0.006	1.429±0.006	1.423±0.006	**1.354±0.013**
kuka2	kukasim	38.94± 0.561	48.93±0.779	60.36±1.005	59.05±1.158	**1.333±0.012**
kuka2	sarcos	18.24±1.599	53.24±6.287	141.7±17.14	211.3±180.2	**1.355±0.014**
kuka2	kuka1	1.461±0.004	1.395±0.004	1.373±0.005	1.374±0.004	**1.331±0.013**

far from train data), and domain-shift (DS) data (removed in-between points), as shown in figure 4, the produced uncertainty estimates are high as test points move away from the train data. Moreover, when DNN predictions match the train data, the produced uncertainty estimates are close to the data noise. We also observe that the patchwork prior mitigates the discontinuity problem of local GPs.

On hyperparameters Next, we examine (i) the influences of hyperparameter choices, and (ii) provide a simple tuning recipe. For this, we examine a regression task using a 5-layered, 200 units MLP on SARCOS robot arm data-set [36]. Here, we vary each hyperparameter while fixing the remaining ones, and we compute the Negative Log Likelihood (NLL). The results are shown in figure 5, with 4 hyperparmeters namely, (i) the number of GP experts, (ii) the subset sizes for clustering, (iii) the pruning level, and (v) the size of the informative data points. We observe the following: (i) the NLL is proportional to the number of GP experts, (ii) the subset size do not significantly influence the NLL, (iii) pruning steps have a tipping point where the NLL increases, and (iv) the active learning parameter has also a tipping point, where the NLL decrease is marginal. In contrary, by the design, the computational complexity increases with (i) lower number of experts and (ii) the size of actively selected points, and decreases with the pruning steps. We find these results confirm our intuition. For example, keeping 2048 GP experts for SARCOS results in 21 data-points per experts on average. This setting is inferior to keeping 32 GP experts, which has 1390 data points on average. Reflecting this, our strategy is to assign more data points for each GP expert, while selecting only the needed active learning points. The pruning levels can then be varied within a range that does not deteriorate the performance in order to reduce the computational complexity of the method.

Evaluation against SOTA baselines Next, we evaluate the performance of our method using robotic datasets namely SARCOS, KUKA1 and KUKA2. The baselines are MC-dropout [3], Laplace Approximation [43, 35], and their fast variants: rMC-dropout [25] and rLaplace [24, 26, 35]. The fast variants are our main competitors: principled Bayesian approaches that are scalable, fast, and training-free, i.e. works with pre-trained DNNs without modifications to the training procedures. We find decoupling DNN training to uncertainty estimates crucial, as it ensures comparisons of uncertainty estimates only by keeping the accuracy of the predictors constant amongst the baselines (in this case, the same pretrained 5-layered MLP across the baselines). Lastly, we adopt zero-shot cross-dataset transfer to systematically evaluate in-domain, OOD and DS scenarios, following the recent insights, that is, we train on a dataset, and evaluate uncertainty estimates for all other datasets.

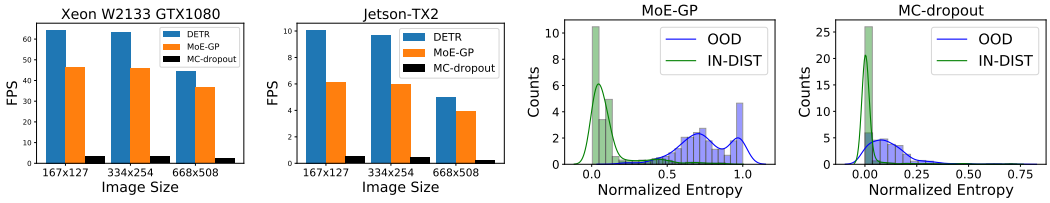

Figure 6: The run-time analysis (left) shows efficiency of our approach when compared to MC-dropout [3] with 20 samples. Any difference to a deterministic DNN can also be seen. The entropy histograms (right) show a novelty detection performance. We note that, parallelizing multiple DNN predictions for a single test input, as in deep ensembles [4] - despite the interests [49], is often not feasible because several large DNNs cannot be stored on a single embedded GPU simultaneously.

The results are in table 1, where we averaged over 3 random initializations and report NLL as a metric (a standard for regression tasks). We find that our method often outperforms other baselines significantly. MC-dropout dominates in 3 experiments. Yet, MC-dropout requires sampling for uncertainty estimates - one of the core problems addressed in our work. We note again, that all the compared methods are training-free, making the comparisons meaningful. Importantly, rLaplace uses the NLM from section 3 to infer Gaussian uncertainty in weight space, in contrast to function space inference with GPs. Thus, the comparison of rLaplace and ours instantiates the comparison of inference between weight space and function space views. We believe that our results show the relevance of GP based uncertainty estimates for DNNs over uncertainty propagation methods.

On scalability Here, we show that (i) our approach scales to KUKA SIM dataset (contains 1984950 data points), and (ii) supports distributed training. Scalability is relevant, as DNNs operate in the regime of big data, and an exact GP does not scale to such settings. Moreover, a key benefit of MoE-GP over other sparse GPs lies in its distributed, local nature, and we can exploit it in practice. The results are depicted in figure 4, where we plot the training time in hours against the number of used GPUs. Using 512 GP experts and 100 iterations for MLL optimization, we find that our method scales to approximately 2 million data points within a day, and less than 8 hours when using 6 GPUs. This experiment shows an evidence that our method scales to large datasets, and validates the claim on distributed training. Our experiment also illustrates the significance of the theoretic results that connects DNNs to MoE-GP. By establishing the theoretic connections between DNNs and sparse GPs, we can obtain methods that possess the strength of both the models in a scalable manner.

On a real robot We perform experiments on a MAV within the context of an on-going space demo mission for future planetary exploration. For this, we train an object detector, DETR [50] with EfficientNet backbone. For testing, we create two scenarios: (i) a carry test, used for training and testing with 1008 manual labels, and (ii) flight tests with different configurations of the objects (e.g. space rover and lander). The later creates OOD samples, with a slight domain shift in data distribution. In field robotics, the assumption that the test set comes from the same training data distribution is routinely violated due to the changes in the environments, and we attempt to create such effects. Moreover, we also evaluate the complexity of our method on a Jetson-TX2, which is used on our MAV. The quantitative results are shown in figure 6, which shows that (i) our method can be fast on a Jetson-TX2, and (ii) can clearly separate OOD samples within this scenario. The accompanying video explains this setup and the qualitative results. More details can be found in the Appendix.

5 Conclusion

The combination of constraints posed by robotic systems makes reliable and fast uncertainty estimation challenging - we need reliable uncertainty estimates as well as efficiency at run-time. In this paper, we propose a principled approach to estimate the predictive uncertainty of neural networks with scalable GPs. Experimentally, we find that our method often leads to superior performance w.r.t the state-of-the art alternatives, scales to large datasets, and provides run-time efficiency on a real robot. Several algorithmic insights are also conveyed through ablation studies and theoretic work. In future, we plan to apply our methodology for an on-going space demo mission for future planetary exploration, and study how we can improve the robustness of the robotic systems in the real world.

References

[1] S. Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

[2] H. Grimmett, R. Triebel, R. Paul, and I. Posner. Introspective Classification for Robot Perception. *International Journal of Robotics Research (IJRR)*, 2015.

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

[4] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017.

[5] 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, 2017.

[6] S. Sun, G. Zhang, J. Shi, and R. Grosse. Functional varitional bayesian neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rkxacs0qY7.

[7] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[8] V. Tresp. Mixtures of gaussian processes. In *Advances in neural information processing systems*, pages 654–660, 2001.

[9] A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

[10] S. Daftry, S. Zeng, J. A. Bagnell, and M. Hebert. Introspective perception: Learning to predict failures in vision systems. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1743–1750. IEEE, 2016.

[11] M. S. Ramanagopal, C. Anderson, R. Vasudevan, and M. Johnson-Roberson. Failing to learn: autonomously identifying perception failures for self-driving cars. *IEEE Robotics and Automation Letters*, 3(4):3860–3867, 2018.

[12] C. Gurău, D. Rao, C. H. Tong, and I. Posner. Learn from experience: probabilistic prediction of perception performance to avoid failure. *The International Journal of Robotics Research*, 37(9): 981–995, 2018.

[13] C. Richter and N. Roy. Safe visual navigation via deep learning and novelty detection. In *Robotics, Science and Systems (RSS)*, 2017.

[14] K. Wong, S. Wang, M. Ren, M. Liang, and R. Urtasun. Identifying unknown instances for autonomous driving. In L. P. Kaelbling, D. Kragic, and K. Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 384–393. PMLR, 30 Oct–01 Nov 2020.

[15] A. Loquercio, M. Segu, and D. Scaramuzza. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters*, 5(2):3153–3160, 2020.

[16] J. Feng, M. Durner, Z.-C. Marton, F. Balint-Benczedi, and R. Triebel. Introspective robot perception using smoothed predictions from bayesian neural networks. In *2019 International Symposium on Robotics Research (ISRR)*, 2019.

[17] A. Harakeh, M. Smart, and S. L. Waslander. Bayesod: A bayesian approach for uncertainty estimation in deep object detectors. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 87–93. IEEE, 2020.

[18] N. Sünderhauf, F. Dayoub, D. Hall, J. Skinner, H. Zhang, G. Carneiro, and P. Corke. A probabilistic challenge for object detection. *Nature Machine Intelligence*, 1(9):443–443, 2019.

[19] D. Winkelbauer, M. Denninger, and R. Triebel. Learning to localize in new environments from synthetic training data. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[20] M. Sensoy, L. Kaplan, and M. Kandemir. Evidential deep learning to quantify classification uncertainty. In *Advances in Neural Information Processing Systems*, pages 3179–3189, 2018.

[21] E. D. Carvalho, R. Clark, A. Nicastro, and P. H. Kelly. Scalable uncertainty for computer vision with functional variational inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12003–12013, 2020.

[22] B. He, B. Lakshminarayanan, and Y. W. Teh. Bayesian deep ensembles via the neural tangent kernel. *arXiv preprint arXiv:2007.05864*, 2020.

[23] B. Adlam, J. Lee, L. Xiao, J. Pennington, and J. Snoek. Exploring the uncertainty properties of neural networks' implicit priors in the infinite-width limit. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=MjvduJCsE4.

[24] D. J. MacKay. Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604, 1992.

[25] J. Postels, F. Ferroni, H. Coskun, N. Navab, and F. Tombari. Sampling-free epistemic uncertainty estimation using approximated variance propagation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2931–2940, 2019.

[26] A. Y. Foong, Y. Li, J. M. Hernández-Lobato, and R. E. Turner. 'in-between'uncertainty in bayesian neural networks. *arXiv preprint arXiv:1906.11537*, 2019.

[27] J. Z. 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. *arXiv preprint arXiv:2006.10108*, 2020.

[28] M. Havasi, R. Jenatton, S. Fort, J. Z. Liu, J. Snoek, B. Lakshminarayanan, A. M. Dai, and D. Tran. Training independent subnetworks for robust prediction. *arXiv preprint arXiv:2010.06610*, 2020.

[29] R. M. Neal. Priors for infinite networks. In *Bayesian Learning for Neural Networks*, pages 29–53. Springer, 1996.

[30] J. Lee, Y. Bahri, R. Novak, S. Schoenholz, J. Pennington, and J. Sohl-dickstein. Deep neural networks as gaussian processes. 2018. URL https://openreview.net/pdf?id=B1EA-M-0Z.

[31] M. E. E. Khan, A. Immer, E. Abedi, and M. Korzepa. Approximate inference turns deep networks into gaussian processes. In *Advances in Neural Information Processing Systems*, pages 3088–3098, 2019.

[32] C. Park and J. Z. Huang. Efficient Computation of Gaussian Process Regression for Large Spatial Data Sets by Patching Local Gaussian Processes. *Journal of Machine Learning Research*, 17(174):1–29, 2016. URL http://jmlr.org/papers/v17/15-327.html.

[33] C. Park and D. Apley. Patchwork kriging for large-scale gaussian process regression. *The Journal of Machine Learning Research*, 19(1):269–311, 2018.

[34] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[35] J. Lee, M. Humt, J. Feng, and R. Triebel. Estimating model uncertainty of neural networks in sparse information form. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5702–5713. PMLR, 13–18 Jul 2020.

[36] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, Jan. 2006.

[37] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016.

[38] S. Gross, M. Ranzato, and A. Szlam. Hard mixtures of experts for large scale weakly supervised vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6865–6873, 2017.

[39] G. Pleiss, J. Gardner, K. Weinberger, and A. G. Wilson. Constant-time predictive distributions for gaussian processes. In *International Conference on Machine Learning*, pages 4114–4123. PMLR, 2018.

[40] Z. Lu, E. Ie, and F. Sha. Uncertainty estimation with infinitesimal jackknife, its distribution and mean-field approximation. *CoRR*, abs/2006.07584, 2020.

[41] J. Wenger, H. Kjellström, and R. Triebel. Non-parametric calibration for classification. *arXiv preprint arXiv:1906.04933*, 2019.

[42] D. J. MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[43] H. Ritter, A. Botev, and D. Barber. A scalable laplace approximation for neural networks. In *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*, volume 6. International Conference on Representation Learning, 2018.

[44] C. Ding and X. He. K-means clustering via principal component analysis. In *Proceedings of the twenty-first international conference on Machine learning*, page 29, 2004.

[45] I. S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–556, 2004.

[46] R. Chitta, R. Jin, T. C. Havens, and A. K. Jain. Approximate kernel k-means: Solution to large scale kernel clustering. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 895–903, 2011.

[47] N. Lawrence, M. Seeger, and R. Herbrich. Fast sparse gaussian process methods: The informative vector machine. In *Proceedings of the 16th annual conference on neural information processing systems*, number CONF, pages 609–616, 2003.

[48] R. Triebel, H. Grimmett, R. Paul, and I. Posner. Driven learning for driving: How introspection improves semantic mapping. In *Robotics Research*, pages 449–465. Springer, 2016.

[49] F. K. Gustafsson, M. Danelljan, and T. B. Schon. Evaluating scalable bayesian deep learning methods for robust computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 318–319, 2020.

[50] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.