Building Bayesian Neural Networks with Blocks: On Structure, Interpretability and Uncertainty

Hao Henry Zhou[1], Yunyang Xiong[2], and Vikas Singh[2,3]

[1]Department of Statistics, University of Wisconsin-Madison
[2]Department of Biostatistics & Medical Informatics, University of Wisconsin-Madison
[3]Department of Computer Sciences, University of Wisconsin-Madison

Abstract

We provide simple schemes to build Bayesian Neural Networks (BNNs), block by block, inspired by a recent idea of computation skeletons. We show how by adjusting the types of blocks that are used within the computation skeleton, we can identify interesting relationships with Deep Gaussian Processes (DGPs), deep kernel learning (DKL), random features type approximation and other topics. We give strategies to approximate the posterior via doubly stochastic variational inference for such models which yield uncertainty estimates. We give a detailed theoretical analysis and point out extensions that may be of independent interest. As a special case, we instantiate our procedure to define a Bayesian *additive* Neural network – a promising strategy to identify statistical interactions and has direct benefits for obtaining interpretable models.

1 Introduction

Bayesian Neural Networks (BNNs) generally refer to a class of algorithms that treat neural network models in a Bayesian manner [15, 13, 19]. Consider the loss function of a well-defined neural network model. In optimizing this loss, one often seeks to find a parameter optimum, say θ^* – a point estimate of the weights. The Bayesian perspective, instead, takes into account the inherent uncertainty in the estimates. To do so, BNNs introduce priors on the network weights: learning then corresponds to approximating the posterior, i.e., $p(\theta|\mathrm{data})$ via probabilistic backpropagation [15], variational inference [13, 12], expectation propagation [17] and so on. When uncertainty estimates are important, as is the case in many applications, BNNs are well suited. On the other hand, approximating the posterior is challenging and further, the choice or design of the prior may not be straightforward.

BNNs are motivated by a probabilistic interpretation of deep neural network learning, which also underlies a related yet distinct body of work known as Deep Gaussian Processes (DGPs). DGPs implement a deep probabilistic *non-parametric model* for compositions of functions: an extension of Gaussian processes, but with a multi-layer structure [10]. These models inherit the expressive power of GPs and provide uncertainty estimates through the posterior distribution, similar to BNNs. But similar to BNNs, calculating the posterior of DGP is difficult. Also, DGPs are based on GPs, so we must solve for the inverse of the kernel matrix: a problem for large datasets. There are recent papers devoted to overcoming the difficulties in training DGPs, which roughly fall in three classes. **First**, one may incorporate the deep neural network structure for designing a "deep" kernel for GP [11, 27]. This offers the benefits of a deep structure but avoids the complexity of a deep GP. The **second** set of methods come up with a specialized design of BNNs [12, 8]: using doubly stochastic variational inference, these results inherit properties of modern innovations in deep learning in that they can use mini-batch training, backpropagation, dropout, and automatic differentiation. The authors in [12, 8] show that such specialized designs of BNNs can actually serve as approximations of DGPs–using specific kernels based on the activation functions of interest. But these ideas, from the DGP point of view, only apply to a limited class of kernels. A **third** line of attack advocates an approximation for the DGP posterior using "inducing points" [5, 9, 14]. In general, this scheme is suitable for *any* kernel, but poses challenges because solving for the inverse of kernel matrix in a deep structure is difficult–therefore, those methods often make strong independence and Gaussianity assumptions. But a recent result in [22] showed – surprisingly – that we do *not* need to force independence or Gaussianity between the layers and the algorithm can, in fact, be trained using mini-batch training, doubly stochastic variational inference and backpropagation for large datasets. Interestingly, we find that the reason mini-batch training, backpropagation, and taking the correlations between layers into account, also common in the second line of work above, can be shown to work in [22] is because the approach actually produces a posterior approximation that belongs to a broad class of BNNs which have a kernel-type structure. This broad class of BNNs have a nice relationship with DGP, maintains correlations between layers and can be trained based on doubly stochastic variational inference. This class which we identify also nicely ties to recent work on deep kernel learning [27], deep random features [11] in the first line of work as well as MC dropout [12] and random feature expansion [8].

Beyond uncertainty estimates: assuming structure on the function class. The discussion above focuses only on uncertainty estimates for the parameters of the model. But in many applications, the interpretability of the model is also important. In statistics, to get an interpretable model, we often impose assumptions on the structure of the function class. For example, an additive structure may pertain to statistical interactions whereas a hierarchical structure could help identify the influence of individual-level effects in a mixed-effects model. In any case, if the model is simple, then making an assumption of structure (e.g., hierarchical, additive), yields two benefits: interpretability *as well as* uncertainty estimates (based on certain distributional assumptions). This suggests that to investigate whether the BNN and DGP results can be made more interpretable, an assumption of structure on the function class may be a good starting point. In fact, within the (non-Bayesian) line of work on DNNs, so-called capsule structures [21] derive representations which respect spatial hierarchies between objects. New results [4] relating multi-resolution analysis (MRA) to CNNs have also appeared [18, 2]. So, can we have strategies for designing deep neural networks which are fundamentally built with *structure* and *interpretability* in mind, but are also easily amenable to *uncertainty estimation*?

Building Bayesian neural networks by blocks. Our goal is to incorporate structural assumptions on BNNs. We describe a procedure based on the so-called "computation skeleton" idea in [11], used to study the relationship between neural networks and kernels. Here, our rationale for using computation skeletons is to design a BNN, block by block, that can efficiently approximate the posterior of DGPs. This scheme offers the flexibility to choose between different structures and/or uncertainty estimation schemes. It retains all useful empirical properties such as mini-batch training, works on large-scale datasets and yields the expressive power of DGPs with kernels. Our results also provide a deeper understanding of the relation between BNN and DGP.

Structural assumption and an example case for interpretability. With the computation skeleton framework for BNN in hand, it will be easy to design approximation schemes for the DGP posterior. While the *uncertainty of prediction accuracy*, has been studied for DNNs [8, 22, 12], another quantity, *statistical interactions*, important for interpretability, has received little attention [26]. If an output depends on several features, one is often interested in changing some features to evaluate how it affects the response. In doing so, we must guarantee that other "uncontrolled" features do *not* influence the response. This confound is called *interaction*: the simultaneous influence of several features on the outcome is *not additive* and the features may jointly affect the outcome. Interpretability means understanding how predictors influence the outcome. But failing to detect statistical interactions causes problems in inferring the features' influence (e.g., the Simpson's paradox). A general network architecture permits *all* features to interact, without the ability to control for the nuisance terms. In statistics, we may use a fully additive statistical model with ANOVA decomposition. Similarly, we propose an additive structure on the network and apply post-training ANOVA decomposition to detect statistical interactions. Describing how the neural network architecture is built, with blocks, additively, is where the computation graph idea is essential – it yields a Bayesian *additive* neural network (BANN). Note that statistical interactions are different from other interpretability focused results in computer vision, specifically on *relevance* or *attribution*[28, 23, 1, 24]. For example, gradient-based methods provide pixel-importance salience maps. While *local* attribution, which most works focus on, describes how the network response changes when we infinitesimally perturb the input sample, *global* attribution captures the marginal effect of a feature on the network output with respect to a baseline. We will show that global attribution can be obtained from our scheme as well.

1.1 Preliminaries

In this section, we briefly review Deep Gaussian process and variational inference schemes that are often used to approximate the posterior distribution to setup the rest of our presentation.

Gaussian processes (GP) and deep Gaussian processes (DGPs). Consider the inference task for a stochastic function $f : \mathbb{R}^p \to \mathbb{R}$, given a likelihood $p(y|f)$ and a set of n observations $\mathbf{y} = (y_1, ..., y_n)^T \in \mathbb{R}^n$ at locations $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_n)^T \in \mathbb{R}^{n\times p}$. We place a GP prior on the function f that models all function values as jointly Gaussian, with a covariance $\mathcal{K} : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$. We use the notation $\mathbf{f} = f(\mathbf{X})$ and $\mathcal{K}(\mathbf{X}, \mathbf{X})_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$. Then, the joint density for $\mathbf{y}$ and $\mathbf{f}$ for a single-layer Gaussian process (GP) is

$$p(\mathbf{y}, \mathbf{f}) = p(\mathbf{f}; \mathbf{X}) \prod_{i=1}^{n} p(y_i|f_i),$$

where $\mathbf{f}|\mathbf{X} \sim N(0, \mathcal{K}(\mathbf{X}, \mathbf{X}))$ and $y_i|f_i \sim N(f_i, \delta^2)$.

Then for L vector-valued stochastic functions denoted as $\mathcal{F}^\ell$, a Deep Gaussian Process (DGP) [10] defines a prior recursively on $\mathcal{F}^1, ..., \mathcal{F}^L$. The prior on each function $\mathcal{F}^\ell$ is an independent GP in each dimension, with input locations given by the function values at the previous layer: the outputs of GPs at layer ℓ are $\{\mathbf{F}^\ell_{.j}\}_{j=1}^{d}$ and the corresponding inputs are $\mathbf{F}^{\ell-1}$. The joint density of the process is

$$p(\mathbf{y}, \{\mathbf{F}^\ell\}_{\ell=1}^{L}) = \prod_{i=1}^{n} p(y_i|f_i^L) \prod_{\ell=1}^{L} p(\mathbf{F}^\ell|\mathbf{F}^{\ell-1}),$$

where $\mathbf{F}^0 = \mathbf{X}$, $\mathbf{F}^\ell \in \mathbb{R}^{n\times d_\ell}$ for $0 < \ell \leq L$. Here, $\mathbf{F}^\ell_{.j}|\mathbf{F}^{\ell-1} \sim N(0, \mathcal{K}^\ell_j(\mathbf{F}^{\ell-1}, \mathbf{F}^{\ell-1}))$ for $1 \leq j \leq d_\ell$, $0 < \ell \leq L$.

Variational inference (VI) for Bayesian models. Consider the joint density of the latent variables $\mathbf{f} = \{f_i\}_{i=1}^{m}$ and the observations $\mathbf{y} = \{y_i\}_{i=1}^{n}$,

$$p(\mathbf{f}, \mathbf{y}) = p(\mathbf{f})p(\mathbf{y}|\mathbf{f}).$$

We know that inference in any Bayesian model amounts to conditioning on the data and computing the posterior $p(\mathbf{f}|\mathbf{y})$. In models like DGP, this calculation is difficult and so, we use approximate inference. A popular strategy is variational inference (VI) [3] which requires specifying a family of *approximate* densities $\mathcal{Q}$. Our goal is to find the member q^* of that family which minimizes the Kullback-Leibler (KL) divergence to the exact posterior,

$$q^*(\mathbf{f}) = \arg \min_{q(\mathbf{f})\in\mathcal{Q}} \mathrm{KL}(q(\mathbf{f})||p(\mathbf{f}|\mathbf{y})).$$

Instead of minimizing the KL divergence, one maximizes the evidence lower bound (ELBO),

$$\text{ELBO}(q) = \mathbb{E}_{q(\mathbf{f})}[\log p(\mathbf{y}|\mathbf{f})] - \text{KL}(q(\mathbf{f}||p(\mathbf{f})).$$

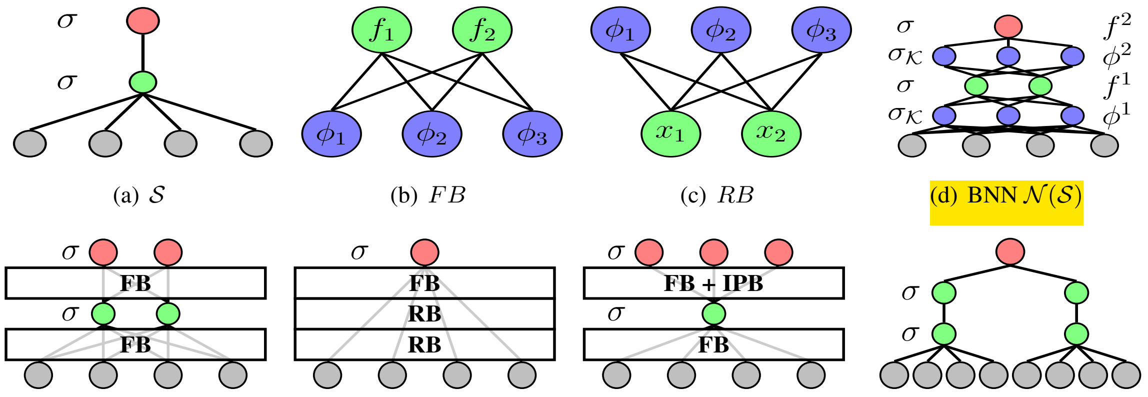
The first term is an expected likelihood, which encourages the densities to place their mass on configurations of the latent variables which explain the observed data. The second term is the negative KL divergence between the variational density and the prior, which encourages densities to lie close to the prior. For DGP type models, VI is a preferred strategy to approximate the posterior.

2 Building Bayesian neural networks: computation skeleton and blocks

We first define the computation skeleton and show how it can lead to a BNN. Then, we show that the constructed BNNs can be seen as a VI approximation for the DGP posterior. Finally, we discuss how to reconstruct other DGP approximations or BNNs [12, 27, 11] in our framework.

What is a computation skeleton? Computation skeleton [11] is a structure to compactly describe a feed-forward computation structure from the inputs to the outputs. Formally, a computation skeleton $\mathcal{S}$ is a multi-layer graph with the bottom nodes representing inputs, the top nodes representing outputs and non-input nodes are labeled by activations σ. In [11], this idea was used to study a family of DNNs and their properties: it was shown that DNNs can be seen as the realization of certain types of structures and their dual kernels. In fact, every $\mathcal{S}$ defines a specific NN structure: Fig. 1(a) shows a two layer fully connected NN, see [11] for more examples. Here, we reuse the name but define a slightly different $\mathcal{S}$ to design BNNs. For notational simplicity, we consider $\mathcal{S}$'s with a single output.

What are blocks? To construct BNNs from a computation skeleton $\mathcal{S}$, we also need two additional components, which we call "blocks". Our first type of block is a **function block** denoted as $FB(\mathbb{P}_\mathbf{v}, r, d)$, which allows every "node" in $\mathcal{S}$ to replicate d times. This will help us in defining Bayesian priors and posteriors. We setup FB as a one layer NN where the inputs nodes and output nodes are fully connected. All incoming edges to the output node f_j as in Fig. 1(b) form a vector $\mathbf{v}_j$. The set of $\mathbf{v}_j$'s for $1 \leq j \leq d$ i.i.d. follow the distribution $\mathbb{P}_\mathbf{v}$ on $\mathbb{R}^r$. $FB(\mathbb{P}_\mathbf{v}, r, d)$ simply takes the inputs $\boldsymbol{\phi} = (\phi_1, ..., \phi_r)^T$ and outputs a d-dimension vector $\mathbf{f}$ with $f_j = \boldsymbol{\phi}^T \mathbf{v}_j$ for $1 \leq j \leq d$. Our second type of block is a **random feature block** denoted as $RB(\mathbb{P}_\mathbf{w}, d, r, \sigma_\mathcal{K})$, which we use to construct random feature approximations for kernels to leverage the expressive power of DGP. We setup RB as a one layer NN with random weights where the inputs nodes and outputs are fully connected. All incoming edges to the output node ϕ_j as in Fig. 1(c) form a vector $\mathbf{w}_j$. The set of $\mathbf{w}_j$'s for $1 \leq j \leq r$ follow the distribution $\mathbb{P}_\mathbf{w}$ on $\mathbb{R}^d$ for $1 \leq j \leq d$. $RB(\mathbb{P}_\mathbf{w}, d, r, \sigma_\mathcal{K})$ takes the inputs $\mathbf{x} = (x_1, ..., x_d)^T$ and outputs a r-dimension vector $\boldsymbol{\phi}$ with $\phi_j = \frac{1}{\sqrt{r}} \sigma_\mathcal{K}(\mathbf{x}^T \mathbf{w}_j)$ for $1 \leq j \leq r$.



(a) $\mathcal{S}$ (b) FB (c) RB (d) BNN $\mathcal{N}(\mathcal{S})$

(e) MC dropout (f) DRF (g) DKL (h) RF, additive

Figure 1: Using $\mathcal{S}$ in (a), one can construct a BNN $\mathcal{N}(\mathcal{S})$ in (d) with the function block (FB) in (b) and the random feature block (RB) in (c). For different $\mathcal{S}$, (e) is MC dropout, (f) is deep random features, (g) is deep kernel learning. (g) also represents a multi-task $\mathcal{S}$ while (h) gives the additive structure.

Constructing a BNN with $\mathcal{S}$, FB and RB blocks. Let us denote s^ℓ to be the number of nodes in layer ℓ of the computation skeleton $\mathcal{S}$. Typically, we may choose $\mathbb{P}_\mathbf{v} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\mathbb{P}_\mathbf{w} \sim \rho N(\mathbf{0}, \mathbf{I})$ for a constant ρ. Alg. 1 shows how given a $\mathcal{S}$, together with FB and RB blocks, we can construct a BNN $\mathcal{N}(\mathcal{S})$ by sequentially replacing edges in $\mathcal{S}$ with a combination of FB and RB from bottom (input nodes) up to the top (output nodes). Shortly, we describe the properties of such a BNN. First, let us see an example. For Fig. 1, using FB in (b) with $r = 3$ and $d = 2$ ($d = 1$ for the last layer) and RB in (c) with $d = 2$ and $r = 3$ ($d = 4$ for the first layer) in Algorithm 1, we construct a BNN in (d) from the in $\mathcal{S}$ in (a). Essentially, we substitute in $FB + RB$ to replace every edge in Fig. 1(a).

Algorithm 1 Constructing a Bayesian neural network (BNN) with computation skeleton and blocks

```
Input: a computation skeleton $\mathcal{S}$. Output: a deep BNN $\mathcal{N}(\mathcal{S})$.
Construct layer 0 in $\mathcal{N}(\mathcal{S})$ by copying inputs (layer 0) from $\mathcal{S}$.
for $\ell = 1$ to $L$ do
    $\mathbf{f}^{\ell-1} = (\mathbf{f}_1^{\ell-1}; ...; \mathbf{f}_{s^{\ell-1}}^{\ell-1}) \in \mathbb{R}^{d^{\ell-1}}$: output vector on layer $\ell-1$ in $\mathcal{N}(\mathcal{S})$.
    For each $\mathbf{f}_j^{\ell-1}$, $1 \le j \le s^{\ell-1}$, apply the activation $\sigma$ in $\mathcal{S}$, and output $\{\sigma(\mathbf{f}_j^{\ell-1})\}_{j=1}^{s^{\ell-1}}$.
    for $i = 1$ to $s^\ell$ do
        $In(i) = \{1 \le j \le s^{\ell-1} |$ if node $j$ in layer $\ell-1$ connects with node $i$ in layer $\ell$ in $\mathcal{S}\}$
        Build $RB(\mathbb{P}_{\mathbf{w}_i^\ell}, d^{\ell-1}, r, \sigma_\mathcal{K})$ on $\{\sigma(\mathbf{f}_j^{\ell-1})\}_{j \in In(i)}$ and output $\boldsymbol{\phi}_i^\ell \in \mathbb{R}^r$.
        Build $FB(\mathbb{P}_{\mathbf{v}_i^\ell}, r, d_i^\ell)$ on $\boldsymbol{\phi}_i^\ell$ and output $\mathbf{f}_i^\ell \in \mathbb{R}^{d_i^\ell}$ in layer $\ell$ of $\mathcal{N}(\mathcal{S})$
```

2.1 Prior and posterior approximation for $\mathcal{N}(\mathcal{S})$

Our remaining task is to describe a prior for $\mathcal{N}(\mathcal{S})$ and then derive a posterior approximation scheme for the construction in Alg. 1. To do so, we define some notations. We use $\mathbf{W}$ for all random weights in the RB blocks, $\mathbf{V}$ gives all BNN weights in the FB blocks and $\mathbf{v}_k^\ell$ denotes the weight vector that goes into kth dimension of $\mathbf{f}^\ell$. The related random features are denoted by $\boldsymbol{\phi}_k^\ell$. For $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_n)^T$, we denote $\mathbf{F}^\ell$ as a matrix with the ith row $\mathbf{F}_{i.}^\ell = \mathbf{f}^\ell(\mathbf{x}_i)$ as the value of $\mathbf{f}^\ell$ evaluated on input $\mathbf{x}_i$. We define $\boldsymbol{\Phi}_k^\ell$ to be the random feature matrix related to $\mathbf{f}_k^\ell$ for $1 \le k \le d^\ell$.

Definition 1. For a BNN $\mathcal{N}(\mathcal{S})$ from Algorithm 1, we treat $\mathbf{W}$ as fixed, then the parameters are only $\mathbf{V}$. We choose $\mathbb{P}_\mathbf{v} = N(\mathbf{0}, \mathbf{I}_r)$ in Algorithm 1 to define the Bayesian prior on $\mathbf{v}_k^\ell$ as $N(\mathbf{0}, \mathbf{I}_r)$ for $1 \le k \le d^\ell$, $1 \le \ell \le L$. This Bayesian prior leads to the relation $p(\mathbf{F}_{.k}^\ell | \mathbf{F}^{\ell-1}) = N(\mathbf{F}_{.k}^\ell; \mathbf{0}, \boldsymbol{\Phi}_k^\ell {\boldsymbol{\Phi}_k^\ell}^T)$, therefore has a distribution over $\{\mathbf{F}^\ell\}_{\ell=1}^L$ which is $p(\{\mathbf{F}^\ell\}_{\ell=1}^L) = \prod_{\ell=1}^L p(\mathbf{F}^\ell | \mathbf{F}^{\ell-1})$.

When the outputs $\mathbf{y}$ and likelihood $p(\mathbf{y}|\mathbf{F}^L)$ are available for the design matrix $\mathbf{X}$, the posterior of BNN $\mathcal{N}(\mathcal{S})$ is intractable. Therefore, we use variational inference to approximate its posterior. We define the variational inference approximation for the posterior of $\mathbf{V}$ in $\mathcal{N}(\mathcal{S})$ by defining the variational posterior q over $\mathbf{V}$ with $\mathbf{v}_k^\ell \sim N(\boldsymbol{\mu}_k^\ell, \boldsymbol{\Sigma}_k^\ell)$. Then, we get the ELBO

$$\text{ELBO} = \sum_{i=1}^n \mathbb{E}_{q(f_i^L)} \log(p(y_i|f_i)) - \text{KL}(q(\mathbf{V})|p(\mathbf{V})) \tag{1}$$

This variational posterior over $\mathbf{V}$ also leads to a posterior over $\{\mathbf{F}^\ell\}_{\ell=1}^L$. We apply a doubly stochastic approximation for the first term in the ELBO, where the sum is estimated using mini-batches and the expectation is approximated with a Monte Carlo sample from the variational posterior $q(f_i^L)$. Both stochastic approximations are unbiased. Further, by reparameterizing $\mathbf{v}_k^\ell = \boldsymbol{\mu}_k^\ell + {\boldsymbol{\Sigma}_k^\ell}^{1/2} N(\mathbf{0}, \mathbf{I}_r)$, the optimization of ELBO can be achieved with mini-batch training and backpropagation[8, 22, 12].

2.2 Relationship of $\mathcal{N}(\mathcal{S})$ to approximate deep Gaussian processes (DGPs)

Having constructed a BNN $\mathcal{N}(\mathcal{S})$ from $\mathcal{S}$, we can study the relationship between $\mathcal{N}(\mathcal{S})$ and DGP. We will show that $\mathcal{N}(\mathcal{S})$ from Alg. 1 is a VI approximation for a DGP posterior. To simplify notation, we assume that all $\{\boldsymbol{\Phi}_k^\ell\}_{k=1}^{d^\ell}$ are the same so we drop the subscript k. We also assume that all $\{d^\ell\}_{\ell=1}^L$ are the same. We define an empirical kernel and its expectation as

$$\begin{aligned} &\text{[Empirical]} \quad \hat{\mathcal{K}}^\ell(\mathbf{f}^{\ell-1}(\mathbf{x}), \mathbf{f}^{\ell-1}(\mathbf{x}')) = \frac{1}{r}\sum_{i=1}^r \sigma_\mathcal{K}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))^T \mathbf{w}_i)\sigma_\mathcal{K}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'))^T \mathbf{w}_i). \\ &\text{[Expectation]} \quad \mathcal{K}^\ell(\mathbf{f}^{\ell-1}(\mathbf{x}), \mathbf{f}^{\ell-1}(\mathbf{x}')) = \mathbb{E}_\mathbf{w} \sigma_\mathcal{K}(\sigma(\mathbf{f}^{\ell-1}(x))^T \mathbf{w})\sigma_\mathcal{K}(\sigma(\mathbf{f}^{\ell-1}(x'))^T \mathbf{w}. \end{aligned} \tag{2}$$

It is easy to check that $\hat{\mathcal{K}}^\ell(\mathbf{f}^{\ell-1}(\mathbf{x}), \mathbf{f}^{\ell-1}(\mathbf{x}')) = \langle \boldsymbol{\phi}^\ell(\mathbf{x}), \boldsymbol{\phi}^\ell(\mathbf{x}') \rangle$. We denote $\hat{\mathcal{K}}^\ell(\mathbf{F}^{\ell-1}, \mathbf{F}^{\ell-1})$ as the $n \times n$ matrix for n inputs. We point out that the prior in Definition 1 is indeed a DGP prior.

Proposition 1. The BNN prior of $\mathcal{N}(\mathcal{S})$ in Def. 1 gives a DGP prior for $\{\mathbf{F}^\ell\}_{\ell=1}^L$. This means that $\mathbf{F}^\ell_{\cdot j}|\mathbf{W}, \mathbf{F}^{\ell-1} \sim N(0, \hat{\mathcal{K}}^\ell(\mathbf{F}^{\ell-1}, \mathbf{F}^{\ell-1}))$, for $1 \leq j \leq d$ and $1 \leq \ell \leq L$.

We can also show that the kernels $\{\hat{\mathcal{K}}^\ell\}_{\ell=1}^L$ for this DGP is close to the kernel $\{\mathcal{K}^\ell\}_{\ell=1}^L$ in (2) if $\sigma_\mathcal{K}$ is ReLU or C-bounded, i.e., $\sigma_\mathcal{K}$ is continuously differentiable and $||\sigma_\mathcal{K}||_\infty, ||\sigma'_\mathcal{K}||_\infty \leq C$.

Theorem 1. If the activation function $\sigma_\mathcal{K}$ is ReLU, then for every $1 \leq \ell \leq L$, on a compact set $\mathcal{M} \in \mathbb{R}^d$ with diameter $diam(\mathcal{M})$ and $\max_{\Delta \in \mathcal{M}} ||\Delta||_2 \leq c_\mathcal{M}$, with probability at least $1 - c_1 c_\mathcal{M} diam(\mathcal{M})^2 \exp\left\{-\frac{r\epsilon^2}{8(1+d)\nu^2_\mathcal{M}}\right\}$,

$$\sup_{\sigma(\mathbf{f}^{\ell-1}(\mathbf{x})), \sigma(\mathbf{f}^{\ell-1}(\mathbf{x}')) \in \tilde{\mathcal{M}}} |\hat{\mathcal{K}}^\ell(\mathbf{f}^{\ell-1}(\mathbf{x}), \mathbf{f}^{\ell-1}(\mathbf{x}')) - \mathcal{K}^\ell(\mathbf{f}^{\ell-1}(\mathbf{x}), \mathbf{f}^{\ell-1}(\mathbf{x}'))| \leq \epsilon,$$

for a constant $c_1 > 0$ and a parameter $\nu_\mathcal{M}$ depending on $\mathcal{M}$. Here, $\tilde{\mathcal{M}}$ specifies that we require $\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))$ and $\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'))$ to be two vectors in $\mathcal{M}$ that are not collinear.

C-boundedness. The uniform concentration bound for C-bounded activation functions is given in the supplement. The C-bounded condition holds for most of the popular sigmoid-like functions such as $1/(1+e^{-x}), erf(x), x/\sqrt{1+x^2}, \tanh(x)$ and $\tan^{-1}(x)$.

Remark 1. In [11], the authors show that the type of kernels constructed from (2) includes linear, polynomial, arc-cosine, radial basis kernels and so on. For ReLU, the authors in [7, 8, 11] point out that $\mathcal{K}^\ell$ is the arc-cosine kernel and [11] shows that $\hat{\mathcal{K}}$ is a sub-exponential random variable.

Since we show that $\{\mathbf{F}^\ell\}_{\ell=1}^L$ in $\mathcal{N}(\mathcal{S})$ can be seen as generated from a DGP based on $\hat{\mathcal{K}}$, we can use many DGP-based approaches to approximate the posterior. Our construction of $\mathcal{N}(\mathcal{S})$ until now is close to the random feature approximation for DGP [8] except that we allow σ to be activation functions (instead of just the identity). Next, we show that using [22] based on inducing points, one also gets the same variational posterior as ours for BNN $\mathcal{N}(\mathcal{S})$. This result enables us to extend our $\mathcal{N}(\mathcal{S})$ construction to be applicable to *any kernel class* and it also implies the underlying connection between random feature [8] and inducing points approximations [22] for DGP. First, we apply the inducing points method in [22] on $\mathcal{N}(\mathcal{S})$ with $\hat{\mathcal{K}}$ to obtain an approximate posterior.

Theorem 2. Using the variational approximation [22] for the posterior of a DGP defined on $\{\hat{\mathcal{K}}^\ell\}_{\ell=1}^L$ with inducing points, we obtain exactly the same variational posterior $q(\{\mathbf{F}^\ell\}_{\ell=1}^L)$ and evidence lower bound ELBO as the variational posterior for $\mathcal{N}(\mathcal{S})$.

The result tells us that the random feature expansion in [8] and inducing points method [22] are *equivalent* for DGPs based on kernels $\{\hat{\mathcal{K}}^\ell\}_{\ell=1}^L$. However, we notice that $\{\mathcal{K}^\ell\}_{\ell=1}^L$ is restricted by $\sigma_\mathcal{K}$ class and does not cover all possible kernels. This issue can be addressed by defining an **inducing points block** IPB to replace RB in Alg. 1. We can show that the derived variational posterior for $\mathcal{N}(\mathcal{S})$ can now be viewed as posterior approximation for a general DGP.

Definition 2. For a kernel $\mathcal{K}$, IPB can be constructed by choosing r additional points $\mathbf{Z}$ (inducing points), taking the inputs $\mathbf{x}$ and outputting an r-dimension vector $\mathcal{K}(\mathbf{x}, \mathbf{Z})\mathcal{K}(\mathbf{Z}, \mathbf{Z})^{-1/2}$.

Theorem 3. Using the variational approximation [22] for the posterior of a DGP defined on $\{\mathcal{K}^\ell\}_{\ell=1}^L$ with inducing points, we can obtain the same variational posterior $q(\{\mathbf{F}^\ell\}_{\ell=1}^L)$ and evidence lower bound ELBO as the variational posterior for $\mathcal{N}(\mathcal{S})$ (with IPB) except a constant offset that does not depend on training (see supplement).

Summary. We see that the main difference between random features [8] and inducing points [22] is that one uses $\mathbf{\Phi}$ and the other uses $\mathcal{K}(\mathbf{x}, \mathbf{Z})\mathcal{K}(\mathbf{Z}, \mathbf{Z})^{-1/2}$ as a rank r basis to approximate the kernel.

2.3 The BNN $\mathcal{N}(\mathcal{S})$ is extremely flexible

We have already shown that the BNN $\mathcal{N}(\mathcal{S})$ from our $\mathcal{S}$ can be seen as approximation for DGP posterior and can be trained efficiently. Now, we show that with a few small changes, interesting special cases emerge. To do so, the changes to Alg. 1, Definition 1 and the variational posterior are,
Change 1) In Alg. 1, inside the inner-most loop, we have one RB (IPB). We allow taking out RB (IPB) entirely or replacing it by multiple sequential RBs (IPBs) as long as they are matched.
Change 2) Earlier, we assumed that the variational posterior q for $\mathbf{v}_i^\ell$ follows a normal distribution. We now allow it to follow a probability mass function and a mixture of two probability mass functions.
Change 3) Earlier in Def. 1, the prior $p(\mathbf{v})$ is a normal distribution. We allow other forms of priors to encourage other types of regularization, such as a Laplace distribution for ℓ_1 sparsity.

Remark. Change 1 allows us to include the classical BNN settings, such as the MC dropout[12] and deep random features concept [11]. Since the exact posterior can be multi-modal, we may want more flexibility beyond normal distribution to approximate it. Therefore, we use Change 2 where the probability mass function results in a standard NN and a mixture of two probability mass function results in MC dropout [12], which are both easy to realize in the optimization of ELBO. For Change 3, when the prior is the normal distribution, the prior for $\mathcal{N}(\mathcal{S})$ can be seen as a DGP prior as we have shown. However, the normal distribution is related to the ℓ_2 regularization from the KL divergence term in the ELBO, while we may need priors in BNNs related to the Lasso or group Lasso type penalties to encourage sparse structure. This intuition motivates Change 3.

Let us see examples of previous works in our framework. **First**, consider the case where do not use any RB in constructing $\mathcal{N}(\mathcal{S})$. This gives us a kernel $\mathcal{K}$ with $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \sigma(\mathbf{x})^T\sigma(\mathbf{x}')$ as in Fig. 1 (e). Further, let us use $q(\mathbf{v})$ as a mixture of two probability mass functions to approximate the posterior. This change leads to MC dropout [12]. **Second**, let us allow multiple RBs in constructing $\mathcal{N}(\mathcal{S})$ as in Fig. 1(f). This ends up representing the deep random feature idea in [11] for GPs. **Third**, in Fig.1 (g), we show another construction that represents deep kernel learning [27], where IPB or RB is not used at all; the variational posterior $q(\mathbf{v})$ is a probability mass function except the last layer.

Remark. Though the framework is flexible, we note that an arbitrary construction can lead to overfitting. Therefore, one still needs to refer to the previous constructions [22, 8, 12, 11, 27] and using our proposal as a guide, consider how to generalize the construction to other $\mathcal{S}$s, how to refine them to obtain more compact forms.

Computation skeleton and structure: As we have emphasized, the computation skeleton captures the most important information of the structure so it helps when we have some structure assumptions for the BNN. In Fig. 1 (g), we see a computation skeleton for multi-task learning where the first layer defines a shared low level function and the second layer defines individual high level functions for each task. In Fig. 1 (h), we have an additive structure where a large neural network is composed by summing several sub neural networks. For those computation skeletons Ss, the construction process for BNN $\mathcal{N}(\mathcal{S})$s directly comes from Alg. 1 and Definition 1.

3 Statistical inference through AddNN: Bayesian additive neural network

Let us see an example of using additive structure in BNNs to detect interactions for interpretability. We specialize our framework to define a Bayesian *additive* neural network. In statistics, given a function f^* between inputs $\mathbf{x} = (x_1, ..., x_p)$ and output y, one can define interaction I_T over a subset of inputs T through ANOVA decomposition $\mathrm{I}_T(\mathbf{x}_T) = \prod_{i\in T}(I_{x_i} - \mathbb{E}_{x_i})\prod_{j\notin T}\mathbb{E}_{x_j} f^*(x_1, ..., x_p)$. We can design an additive neural network (AddNN) to partially represent the ANOVA decomposition: $f(\mathbf{x}) = \sum_{j=1}^{k} g_j(\mathbf{x})$, where every g_j is a NN with the first layer regularized by group Lasso type penalty. We can use post-training ANOVA decomposition (replace $\mathbb{E}$ in ANOVA by the empirical expectation $\mathbb{E}^n$ with n samples) to measure the interactions:

$$\mathrm{I}_T^n(\mathbf{x}_T) = \prod_{i\in T}(I_{x_i} - \mathbb{E}^n_{x_i})\prod_{j\notin T}\mathbb{E}^n_{x_j} f(x_1, ..., x_p). \tag{3}$$

We have the following theorem for the complexity of calculating this measure for AddNN,

Theorem 4. If there exist inputs clusters $\{T_j^*\}_{j=1}^{k^*}$ such that $f^*(\mathbf{x}) = \sum_{j=1}^{k^*} g_j^*(\mathbf{x}_{T_j^*})$ with k^* of the order of a polynomial in p and $c = \max_{j=1}^{k^*} |T_j^*| = O(\log p)$, then there exists a trained AddNN that predicts $\mathbf{y}$ well and restricts the number of possible interactions to be at most a polynomial in p. Further, if every sub neural network has L layers with d hidden units, then the complexity of (3) is at most $n^c k^* d^{2L-1}$, which is also polynomial in p.

The result does not hold for an arbitrary NN, which has 2^p possible interactions. The complexity of (3) is $n^p(k^*d)^{2L-1}$, exponential in p. AddNN is far more efficient when f^* has additive structure.

Uncertainty in AddNN: We show the additive neural network computation skeleton as an example in Fig. 1 (h). Then, we can easily construct the Bayesian formulation of AddNN with various uncertainty estimates methods. We only require a specific design of the first layer for variable selection. For every sub-neural network, the first layer is only built with FB and the prior on the weights is $p(\mathbf{V}^1) \sim \exp(-\sum_{i=1}^{p} ||\mathbf{v}_i^1||_2)$ where $\mathbf{v}_i^1$ refers to the weight vector emanating from the ith input. The variational posterior $q(\mathbf{V}^1)$ is the probability mass function to make top layers stable.

4 Experiments

We first evaluate the performance of our AddNN model on synthetic experiments for regression and interaction detection for additive functions. Then, we use Alg. 1 to construct four different types of AddNNs (where each provides uncertainty estimates) and check its utility for prediction and identifying interaction strength. Finally, we show how AddNN can infer main effects and statistical interactions of features with uncertainties for interpretability. Further, we use AddNN on eight benchmark datasets used in existing papers to show that our model offers competitive results.

AddNN, BNN, BART and NID on prediction accuracy and interaction detection. We compare AddNN with BNN (with a single neural network), BART (Bayesian additive regression tree) and NID (Neural interaction detection) in terms of prediction accuracy and interaction detection. For AddNN, we use the setup in § 3, where the group Lasso penalty is applied on the first layer. We use 10 compact sub-NNs for AddNN and a single (but more complex) neural network for BNN (see supplement). For BART and NID, we use the setup in [6, 26]. Both AddNN and BNN here are based on the MC dropout type construction (see § 2). First, we compare RMSE (root mean-squared-error). We run 4 synthetic experiments using the functions in the left Tab. 1. For every experiment, we use one function f in the left Tab. 1 to generate 5000 train/test samples (10 features, 1 response), where for every input $\mathbf{x}$, each dimension of the inputs are i.i.d. generated from the uniform distribution on $(0, 1]$ and the response y is $y = f(\mathbf{x}) + \epsilon$, with $\epsilon \sim N(0, 1)$. From Tab. 2, we see that the AddNN yields comparable (and sometimes better) RMSE compared to baselines. Though the prediction performance is similar, note that AddNN is a much more compact design: AddNN has just ∼500 edges while the BNN has 7000 edges, NID has 2000 edges and BART has 200 trees (see Tab. 2).
Next, we compare AddNN and NID for interaction detection (other two baselines are not applicable). To detect interactions, AddNN first calculates the interaction functions from (3), then their empirical ℓ_2 norms are used as the "interaction strength", and then AddNN selects the top k interactions. Possible interaction candidates are based on the group-Lasso clusters for every "sub-NN" in our additive model. For NID, we use the setup in [26]. We run the same experiments as the RMSE setting using left Tab. 1. To assess ranking

Table 1: (left) Synthetic functions used in our experiments, based on [26];(right) Average test performance in RMSE for AddNN(ours) and MC dropout on benchmarks.

Method	Formula
f_1	$10\sin(\pi x_1 x_2) + 20(x_3 - .5)^2$ $+10x_4 + 5x_5$
f_2	$10\exp(x_1 x_2) - 20\cos(x_3 + x_4 + x_5)$ $+7\arcsin(x_9 x_{10})$
f_3	$\exp(\lvert x_1 x_2\rvert + 1) + \exp(\lvert x_3 + x_4\rvert + 1)$ $-19\cos(x_5 + x_6) - 10\sqrt{x_8^2 + x_9^2 + x_{10}^2}$
f_4	$\frac{1}{1+x_1^2+x_2^2+x_3^2} - 5\sqrt{\exp(x_4 + x_5)}$ $+10\lvert x_6 + x_7\rvert + 6x_8 x_9 x_{10}$

Measure	RMSE (AddNN)	RMSE ([12])
Boston	3.03 ± 0.12	2.97 ± 0.19
Concrete	5.18 ± 0.14	5.23 ± 0.12
Energy	0.65 ± 0.03	1.66 ± 0.04
Kin8nm	0.07 ± 0.00	0.10 ± 0.00
Naval	0.01 ± 0.00	0.01 ± 0.00
Power	4.04 ± 0.03	4.02 ± 0.04
Protein	4.07 ± 0.01	4.36 ± 0.01
Wine	0.66 ± 0.01	0.62 ± 0.01

Table 2: Comparisons between AddNN, BNN, BART and NID. BNN and BART do not detect interactions.

	RMSE				Top rank recall (noise, $\sigma^2 = 1, 3, 5$)					
	AddNN ($0.5k$)	BNN ($7k$)	BART	NID ($20k$)	**Ours** (AddNN)			NID		
f_1	1.07 ± 0.01	1.15 ± 0.01	1.07 ± 0.01	1.09 ± 0.01	**1**	**1**	**1**	1	1	1
f_2	1.16 ± 0.01	1.22 ± 0.02	1.43 ± 0.02	1.44 ± 0.02	**1**	**1**	**2/3**	1	2/3	0
f_3	1.35 ± 0.01	1.32 ± 0.01	1.24 ± 0.02	1.42 ± 0.02	3/4	**3/4**	**2/4**	1	2/4	1/4
f_4	1.13 ± 0.01	1.13 ± 0.01	1.17 ± 0.01	1.40 ± 0.02	**3/4**	**3/4**	**2/4**	2/4	2/4	1/4

quality, we use the top-rank recall metric [26]: a recall of interaction rankings where only those interactions that are correctly ranked before we encounter any false positives are considered. Only one superset interaction from each sub-function of f is counted as a true interaction. From Tab. 2, we see that the AddNN outperforms NID for interaction detection.

Four different types of AddNN. As described in § 2, we can derive other uncertainty schemes using AddNN. Then, we can calculate uncertainty based on each of these schemes. Here, we use the mean log likelihood (MLL) and the empirical ℓ_2 norm of the interaction or main effect function from (3). The empirical ℓ_2 norm measures the strength of the interaction or main effect. We calculate the uncertainty for AddNN but do not compare with NID [26] since it cannot model uncertainty. Here, we show an example for f_1 from left Tab. 1. Tab. 3 shows that that all four methods (derivable from our proposal) correctly yields interactions between x_1 and x_2 as well as the main effects. AddNN provides predictions *and* interactions with uncertainties.

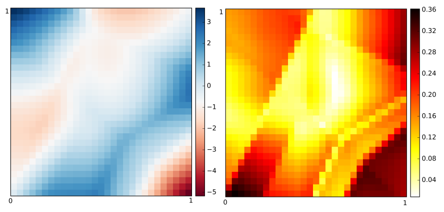


Figure 2: Interaction between x_1 and x_2 for f_1 in Tab. 1. Left (and right) image shows the mean interaction (and its standard deviation).

Table 3: Constructing multiple types of uncertainty estimates for AddNN of f_1.

Measure	MLL	Interaction	Main effect				
		$(1, 2)$	1	2	3	4	5
MC dropout	-1.61 ± 0.09	1.51 ± 0.05	2.44 ± 0.15	2.35 ± 0.10	1.69 ± 0.06	3.18 ± 0.04	1.63 ± 0.03
RF	-1.60 ± 0.09	1.52 ± 0.06	2.39 ± 0.08	2.31 ± 0.11	1.70 ± 0.11	3.19 ± 0.06	1.61 ± 0.04
DKL	-1.53 ± 0.08	1.59 ± 0.04	2.44 ± 0.23	2.32 ± 0.13	1.70 ± 0.08	3.16 ± 0.06	1.61 ± 0.04
DRF	-1.56 ± 0.07	1.40 ± 0.02	2.36 ± 0.05	2.35 ± 0.10	1.71 ± 0.03	3.15 ± 0.03	1.59 ± 0.02

Uncertainty and interpretability using AddNN. We show one representative example showing how (3) with our formulation can be used to model the interaction between x_1 and x_2 for f_1 in Tab. 1. We plot the average interaction function and the uncertainty function in Fig. 2 as a heatmap – this ability is rarely available for deep neural network models and can be very useful for interpretability.

Benchmark experiments. Finally, we apply AddNN on common datasets used by other authors [12, 22]. Here, as shown in right Tab. 1, we find that AddNN (which is a more compact model) yields competitive performance in addition to the other features it natively provides such as interaction (interpretability) and uncertainty discussed above. This implies that these additional benefits do not come at a cost of performance. The complete table with previous works are in supplement.

5 Discussion

We presented a scheme by adapting the computation skeleton idea to construct BNNs. Our models can be trained using modern innovations including mini-batch training, dropout, and automatic differentiation. We showed that a broad class of BNNs, realized by our framework, ties nicely to DGPs, deep kernel learning, MC dropout and other topics. As a special case, we proposed an Bayesian additive neural network that competes favorably with state-of-the-arts methods and provides uncertainty and interpretability, via statistical interactions.

Acknowledgments We thank Grace Wahba and Ronak Mehta for suggestions and ideas. This work is supported by NIH Grants R01 AG040396, R01 EB022883, UW CPCP AI117924, and R01 AG021155, and NSF Awards DMS 1308877 and CAREER 1252725. We received partial support from NIH Grants UW ADRC AG033514 and UW ICTR 1UL1RR025011 and funding from a University of WisconsinMadison/German Center for Neurodegenerative Diseases collaboration initiative.

References

[1] Marco Ancona, Enea Ceolini, Cengiz ztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning*

Representations, 2018.

[2] Toms Angles and Stphane Mallat. Generative networks as inverse problems with scattering transforms. In *International Conference on Learning Representations*, 2018.

[3] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[4] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

[5] Thang Bui, Daniel Hernández-Lobato, Jose Hernandez-Lobato, Yingzhen Li, and Richard Turner. Deep gaussian processes for regression using approximate expectation propagation. In *International Conference on Machine Learning*, pages 1472–1481, 2016.

[6] Hugh A Chipman, Edward I George, Robert E McCulloch, et al. Bart: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.

[7] Youngmin Cho and Lawrence K Saul. Kernel methods for deep learning. In *Advances in neural information processing systems*, pages 342–350, 2009.

[8] Kurt Cutajar, Edwin V Bonilla, Pietro Michiardi, and Maurizio Filippone. Random feature expansions for deep gaussian processes. In *International Conference on Machine Learning*, pages 884–893, 2017.

[9] Zhenwen Dai, Andreas Damianou, Javier González, and Neil Lawrence. Variational auto-encoded deep gaussian processes. *arXiv preprint arXiv:1511.06455*, 2015.

[10] Andreas Damianou and Neil Lawrence. Deep gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215, 2013.

[11] Amit Daniely, Roy Frostig, and Yoram Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In *Advances In Neural Information Processing Systems*, pages 2253–2261, 2016.

[12] 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.

[13] Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pages 2348–2356, 2011.

[14] James Hensman and Neil D Lawrence. Nested variational compression in deep gaussian processes. *arXiv preprint arXiv:1412.1370*, 2014.

[15] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

[16] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

[17] Pasi Jylänki, Aapo Nummenmaa, and Aki Vehtari. Expectation propagation for neural networks with sparsity-promoting priors. *The Journal of Machine Learning Research*, 15(1):1849–1901, 2014.

[18] Stéphane Mallat. Understanding deep convolutional networks. *Phil. Trans. R. Soc. A*, 374(2065):20150203, 2016.

[19] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[20] Mark Rudelson. Invertibility of random matrices: norm of the inverse. *Annals of Mathematics*, pages 575–600, 2008.

[21] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3859–3869, 2017.

[22] Hugh Salimbeni and Marc Deisenroth. Doubly stochastic variational inference for deep gaussian processes. In *Advances in Neural Information Processing Systems*, pages 4591–4602, 2017.

[23] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153, 2017.

[24] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328, 2017.

[25] Terence Tao. *Topics in random matrix theory*, volume 132. American Mathematical Soc., 2012.

[26] Michael Tsang, Dehua Cheng, and Yan Liu. Detecting statistical interactions from neural network weights. In *International Conference on Learning Representations*, 2018.

[27] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378, 2016.

[28] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

6 Supplement

In this supplement, we first discuss the extensions of the model to multiple outputs and classification and discuss how to incorporate the bias terms in NN. Then we show the proofs for the theorems in the main body. Finally, we present details for models used in experiments and provide more experiment results.

6.1 The extension to multiple outputs, classification and including bias terms

For the output $\mathbf{y} \in \mathbb{R}^d$, we permit our computation skeleton to have d output nodes as well. Then after we run our construction algorithm, we obtain a BNN with d outputs $\mathbf{f}^L$ at the last layer. Then the analysis and properties for the single output case also hold for the multiple output case.

In regression task, we assume the likelihood $p(\mathbf{y}|\mathbf{F}^L)$ of the output $\mathbf{y}$ to be a normal distribution, given the input matrix $\mathbf{X}$ with n samples and the relevant output $\mathbf{F}^L$ at the last layer of BNN. We output $\mathbf{F}^L$ to estimate the mean of $\mathbf{y}$. The relevant loss in the optimization of ELBO is the mean square loss. This is usually considered for the regression task. In a classification task with $\mathbf{y} \in \{0, 1, ..., k\}$ in k categories, we assume the likelihood $p(y_i|\mathbf{F}^L_{i.}) = \frac{\exp(\mathbf{F}^L_{iy_i})}{\sum_{j=1}^k \exp(\mathbf{F}^L_{ij})}$ for $1 \leq i \leq n$. Then the relevant loss in the optimization of ELBO is

$$\frac{1}{n}\sum_{i=1}^{n} \log\left(\frac{\exp(\mathbf{F}^L_{iy_i})}{\sum_{j=1}^k \exp(\mathbf{F}^L_{ij})}\right).$$

We can add bias terms into the framework. The bias term with random weight can either be incorporated into the construction of RB [11] or be treated as a parameter in the BNN [12]. We refer one to see these two works for incorporating bias terms.

Remark. The statements for the extension to multiple outputs and classification hold for DGPs as well.

6.2 Proofs for theorems in the main body

In this section, we give the proofs for theorems in the main body.

6.2.1 The proof for the relation between activation functions and kernels

First, we prove theorems on the relation between activation functions and kernels.

The uniform concentration bound for C-bounded activation functions and its proof

First, we present the uniform concentration bound for C-bounded activation functions and its proof.

Theorem 0. If the activation function $\sigma_{\mathcal{K}}$ is C-bounded, meaning it is continuously differentiable and $||\sigma_{\mathcal{K}}||_\infty, ||\sigma'_{\mathcal{K}}||_\infty \leq C$, then for every $1 < \ell \leq L$, on a compact set $\mathcal{M} \in \mathbb{R}^d$ with diameter $diam(\mathcal{M})$, with probability at least $1 - c_1 diam(\mathcal{M})^2 \exp\left\{-\frac{\epsilon^2 r}{8(1+d)C}\right\}$,

$$\sup_{\sigma(\mathbf{f}^{\ell-1}(\mathbf{x})),\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'))\in\mathcal{M}} |\hat{\mathcal{K}}^{\ell}(\mathbf{f}^{\ell-1}(\mathbf{x}),\mathbf{f}^{\ell-1}(\mathbf{x}')) - \mathcal{K}^{\ell}(\mathbf{f}^{\ell-1}(\mathbf{x}),\mathbf{f}^{\ell-1}(\mathbf{x}'))| \leq \epsilon,$$

for a constant $c_1 > 0$.

Proof. (a) For a C-bounded activation function, since $||\sigma_{\mathcal{K}}(\cdot)||_\infty \leq C$, for fixed $\mathbf{f}^{\ell-1}(\mathbf{x})$ and $\mathbf{f}^{\ell-1}(\mathbf{x}')$, the r random variables $\{\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))^T\mathbf{w}_i)\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'))^T\mathbf{w}_i)\}_{i=1}^r$ are independent and lie in a bounded interval $[-C, C]$. Then using Hoeffdings' inequality, we get that

$$\mathbb{P}(|\hat{\mathcal{K}}^{\ell}(\mathbf{f}^{\ell-1}(\mathbf{x}),\mathbf{f}^{\ell-1}(\mathbf{x}')) - \mathcal{K}^{\ell}(\mathbf{f}^{\ell-1}(\mathbf{x}),\mathbf{f}^{\ell-1}(\mathbf{x}'))| \geq \epsilon) \leq 2\exp(-\frac{2r\epsilon^2}{4C^2}). \quad (4)$$

Next we show that for a compact set $\mathcal{M}$ of $\mathbb{R}^d$ with diameter $diam(\mathcal{M})$, with probability at least $1 - 2^{11}\left(\frac{C^4 d\ diam(\mathcal{M})^2}{\epsilon^2 r}\right)^{\frac{d}{1+d}} \exp\left\{-\frac{r\epsilon^2}{8(1+d)C^2}\right\}$,

$$\sup_{\sigma(\mathbf{f}^{\ell-1}(\mathbf{x})),\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'))\in\mathcal{M}} |\hat{\mathcal{K}}^{\ell}(\mathbf{f}^{\ell-1}(\mathbf{x}),\mathbf{f}^{\ell-1}(\mathbf{x}')) - \mathcal{K}^{\ell}(\mathbf{f}^{\ell-1}(\mathbf{x}),\mathbf{f}^{\ell-1}(\mathbf{x}'))| \leq \epsilon.$$

Since $\mathcal{M}$ has diameter $diam(\mathcal{M})$, we can find δ-net that covers $\mathcal{M}$ using at most $T = (4diam(\mathcal{M})/\delta)^d$ balls of radius δ. Let $\{\Delta_i\}_{i=1}^T$ denote the centers of these balls. Then using (4) and union bounds, for any two centers, such as Δ_1 and Δ_2, with probability at least $1 - 2\exp(\log(T^2) - \frac{2r\epsilon^2}{16C^2})$,

$$|\hat{\mathcal{K}}^{\ell}(\Delta_1,\Delta_2) - \mathcal{K}^{\ell}(\Delta_1,\Delta_2)| \leq \frac{\epsilon}{2}. \quad (5)$$

For the function $\mathbf{u}(\mathbf{x},\mathbf{x}') = \hat{\mathcal{K}}^{\ell}(\mathbf{f}^{\ell-1}(\mathbf{x}),\mathbf{f}^{\ell-1}(\mathbf{x}')) - \mathcal{K}^{\ell}(\mathbf{f}^{\ell-1}(\mathbf{x}),\mathbf{f}^{\ell-1}(\mathbf{x}'))$, we have the inequality from partial derivative that

$$|\mathbf{u}(\mathbf{x},\mathbf{x}') - \mathbf{u}(\mathbf{x}_0,\mathbf{x}_0')| \leq L_{\sigma_{\mathcal{K}}}(||\sigma(\mathbf{f}^{\ell-1}(\mathbf{x})) - \sigma(\mathbf{f}^{\ell-1}(\mathbf{x}_0))||_2 + ||\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}')) - \sigma(\mathbf{f}^{\ell-1}(\mathbf{x}_0'))||_2), \quad (6)$$

where

$$\begin{aligned}
L_{\sigma_{\mathcal{K}}} &= \arg\max_{\sigma(\mathbf{f}^{\ell-1}(\mathbf{x})),\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'))\in\mathcal{M}} ||\frac{1}{r}\sum_{i=1}^r \frac{\partial\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))^T\mathbf{w}_i)}{\partial\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))}\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'))^T\mathbf{w}_i) \\
&- \mathbb{E}_{\mathbf{w}}\frac{\partial\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))^T\mathbf{w})}{\partial\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))}\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'))^T\mathbf{w})||_2 \\
&= ||\frac{1}{r}\sum_{i=1}^r \frac{\partial\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}^*))^T\mathbf{w}_i)}{\partial\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))}\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'^*))^T\mathbf{w}_i) \\
&- \mathbb{E}_{\mathbf{w}}\frac{\partial\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}^*))^T\mathbf{w})}{\partial\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))}\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'^*))^T\mathbf{w})||_2.
\end{aligned}$$

We also have that

$$\begin{aligned}
\mathbb{E}L_{\sigma_\mathcal{K}}^2 &= \mathbb{E}||\frac{1}{r}\sum_{i=1}^{r}\frac{\partial\sigma_\mathcal{K}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}^*))^T\mathbf{w}_i)}{\partial\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))}\sigma_\mathcal{K}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x'}^*))^T\mathbf{w}_i) \\
&- \mathbb{E}_\mathbf{w}\frac{\partial\sigma_\mathcal{K}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}^*))^T\mathbf{w})}{\partial\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))}\sigma_\mathcal{K}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x'}^*))^T\mathbf{w})||_2^2 \\
&= \mathbb{E}||\frac{1}{r}\sum_{i=1}^{r}\frac{\partial\sigma_\mathcal{K}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}^*))^T\mathbf{w}_i)}{\partial\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))}\sigma_\mathcal{K}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x'}^*))^T\mathbf{w}_i)||_2^2 \\
&- ||\mathbb{E}_\mathbf{w}\frac{\partial\sigma_\mathcal{K}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}^*))^T\mathbf{w})}{\partial\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))}\sigma_\mathcal{K}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x'}^*))^T\mathbf{w})||_2^2 \\
&= \frac{1}{r^2}\sum_{i=1}^{r}\mathbb{E}||\frac{\partial\sigma_\mathcal{K}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}^*))^T\mathbf{w}_i)}{\partial\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))}\sigma_\mathcal{K}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x'}^*))^T\mathbf{w}_i)||_2^2 \\
&- \frac{1}{r^2}||\mathbb{E}_\mathbf{w}\frac{\partial\sigma_\mathcal{K}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}^*))^T\mathbf{w})}{\partial\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))}\sigma_\mathcal{K}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x'}^*))^T\mathbf{w})||_2^2 \\
&\leq \frac{1}{r^2}C^4\sum_{i=1}^{r}\mathbb{E}||\mathbf{w}_i||_2^2 \\
&= \frac{C^4 d}{r}.
\end{aligned}$$

Therefore, by Markov's inequality,

$$\mathbb{P}(L_{\sigma_\mathcal{K}} \geq \frac{\epsilon}{4\delta}) \leq \mathbb{E}L_{\sigma_\mathcal{K}}^2\frac{16\delta^2}{\epsilon^2} \leq \frac{16\delta^2 C^4 d}{\epsilon^2 r}$$

Then using Eq. (6), with probability at least $1 - \frac{16\delta^2 C^4 d}{\epsilon^2 r}$,

$$|u(\mathbf{x}, \mathbf{x}') - u(\mathbf{x}_0, \mathbf{x}_0')| \leq \frac{\epsilon}{2}$$

This inequality combined with Eq. (5) enables us to conclude that

$$\sup_{\sigma(\mathbf{f}^{\ell-1}(\mathbf{x})),\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'))\in\mathcal{M}} |\hat{\mathcal{K}}^\ell(\mathbf{f}^{\ell-1}(\mathbf{x}), \mathbf{f}^{\ell-1}(\mathbf{x}')) - \mathcal{K}^\ell(\mathbf{f}^{\ell-1}(\mathbf{x}), \mathbf{f}^{\ell-1}(\mathbf{x}'))| \leq \epsilon.$$

with probability at least $1 - \frac{16\delta^2 C^4 d}{\epsilon^2 r} - 2\exp(\log(T^2) - \frac{2r\epsilon^2}{16C^2})$. Recall that $T = (4diam(\mathcal{M})/\delta)^d$, so the probability has a format of $1 - \kappa_1\delta^2 - \kappa_2\delta^{-2d}$ for δ. By setting $\delta = \frac{\kappa_2}{\kappa_1}^{\frac{1}{2+2d}}$, we have the probability as $1 - 2\kappa_1^{\frac{2d}{2+2d}}\kappa_2^{\frac{2}{2+2d}}$. So the probability is at least

$$1 - 2^{11}\left(\frac{C^4 d\, diam(\mathcal{M})^2}{\epsilon^2 r}\right)^{\frac{d}{1+d}} \exp\left\{-\frac{r\epsilon^2}{8(1+d)C^2}\right\}$$

□

Proof of Theorem 1 for ReLU

We have seen how to control the distance between empirical kernel and the expectation kernel uniformly for C-bounded activation functions, now we present the proof for ReLU activation functions.

Theorem 1. If the activation function $\sigma_{\mathcal{K}}$ is ReLU, then for every $1 \leq \ell \leq L$, on a compact set $\mathcal{M} \in \mathbb{R}^d$ with diameter $diam(\mathcal{M})$ and $\max_{\Delta \in \mathcal{M}} ||\Delta||_2 \leq c_{\mathcal{M}}$, with probability at least $1 - c_1 c_{\mathcal{M}} diam(\mathcal{M})^2 \exp\left\{-\frac{r\epsilon^2}{8(1+d)\nu_{\mathcal{M}}^2}\right\}$,

$$\sup_{\sigma(\mathbf{f}^{\ell-1}(\mathbf{x})), \sigma(\mathbf{f}^{\ell-1}(\mathbf{x}')) \in \tilde{\mathcal{M}}} |\hat{\mathcal{K}}^{\ell}(\mathbf{f}^{\ell-1}(\mathbf{x}), \mathbf{f}^{\ell-1}(\mathbf{x}')) - \mathcal{K}^{\ell}(\mathbf{f}^{\ell-1}(\mathbf{x}), \mathbf{f}^{\ell-1}(\mathbf{x}'))| \leq \epsilon,$$

for a constant $c_1 > 0$ and a parameter $\nu_{\mathcal{M}}$ depending on $\mathcal{M}$. Here, $\tilde{\mathcal{M}}$ specifies that we require $\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))$ and $\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'))$ to be two vectors in $\mathcal{M}$ that are not collinear.

Proof. For the ReLU activation $\sigma_{\mathcal{K}}(x) = \max(0, x)$, we use concentration bound for sub-exponential random variable to show the result for fixed points. We define $u = \sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))^T \mathbf{w}) \sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'))^T \mathbf{w})$. Our first goal is to compute $\mathbb{E}_{\mathbf{w}}[e^{\lambda \mathbf{u}}]$. Since $\mathbf{w}$ follows a normal distribution that is symmetric, how we choose axis does not influence the results. Therefore, we choose axis such that $\sigma(\mathbf{f}^{\ell-1}(\mathbf{x})) = \mathbf{e}_1 ||\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))||_2$ and $\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}')) = (\mathbf{e}_1 \cos\theta + \mathbf{e}_2 \sin\theta) ||\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'))||_2$ where $\mathbf{e}_1$ and $\mathbf{e}_2$ refer to standard vector for the first and second axis. We denote $C_f = ||\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))||_2 ||\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'))||_2 \leq c_{\mathcal{M}}^2$.

$$\mathbb{E}_{\mathbf{w}}[e^{\lambda u}] = \frac{1}{2\pi} \int_{-\infty}^{\infty} dw_1 \int_{-\infty}^{\infty} dw_2 e^{-\frac{1}{2}(w_1^2 + w_2^2)} e^{\lambda C_f \max(0, w_1) \max(0, w_1 \cos\theta + w_2 \sin\theta)}.$$

We switch (w_1, w_2) by $(\tilde{w}_1, \tilde{w}_2) = (w_1, w_1 \cos\theta + w_2 \sin\theta)$, then we get that

$$\mathbb{E}_{\mathbf{w}}[e^{\lambda u}] = \frac{1}{2\pi \sin\theta} \int_0^{\infty} d\tilde{w}_1 \int_0^{\infty} d\tilde{w}_2 e^{-\frac{\tilde{w}_1^2 + \tilde{w}_2^2 - 2\tilde{w}_1 \tilde{w}_2 \cos\theta}{2 \sin\theta^2}} e^{\lambda C_f \tilde{w}_1 \tilde{w}_2}.$$

We switch $(\tilde{w}_1, \tilde{w}_2)$ by $(\tilde{r}, \tilde{\phi})$ with $\tilde{w}_1 = \tilde{r} \sin\tilde{\phi}$ and $\tilde{w}_2 = \tilde{r} \cos\tilde{\phi}$, then we get that

$$\mathbb{E}_{\mathbf{w}}[e^{\lambda u}] = \frac{1}{2\pi \sin\theta} \int_0^{\pi/2} d\tilde{\phi} \int_0^{\infty} \tilde{r} d\tilde{r} e^{-\tilde{r}^2 \frac{1 - \sin 2\tilde{\phi} \cos\theta}{2 \sin\theta^2}} e^{\tilde{r}^2 \frac{\lambda C_f \sin 2\tilde{\phi}}{2}}.$$

Through the known mean calculation of half normal distribution that

$$\frac{a\sqrt{2}}{\sqrt{\pi}} = \int_{x \geq 0} x dx \frac{\sqrt{2}}{a\sqrt{\pi}} \exp(-\frac{x^2}{2a^2}),$$

for any a, we know that

$$a^2 = \int_{x \geq 0} x dx \exp(-\frac{x^2}{2a^2}),$$

for any a. We use this relation to calculate the integral of $\tilde{r}$ and we get that

$$\mathbb{E}_{\mathbf{w}}[e^{\lambda u}] = \frac{1}{2\pi \sin\theta} \int_0^{\pi/2} d\tilde{\phi} \frac{\sin\theta^2}{1 - \sin 2\tilde{\phi}\cos\theta - \lambda C_f \sin 2\tilde{\phi} \sin\theta^2}$$
$$= \frac{\sin\theta}{2\pi} \int_0^{\pi/2} d\alpha \frac{1}{1 - \cos\alpha\cos\theta - \lambda C_f \cos\alpha \sin\theta^2},$$

by switching $\tilde{\phi}$ to $\alpha = 2\tilde{\phi} - \frac{\pi}{2}$. It is known that

$$\int_0^{\xi} d\alpha \frac{1}{1 - \cos\alpha\cos\theta} = \frac{1}{\sin\theta} \tan^{-1}\left(\frac{\sin\theta\sin\xi}{\cos\xi - \cos\theta}\right),$$

which can be verified by calculating the derivative of the right side [7]. Therefore, by setting $\xi = \frac{\pi}{2}$,

$$\int_0^{\pi} 2d\alpha \frac{1}{1 - \cos\alpha\cos\theta} = \frac{\pi - \theta}{\sin\theta}.$$

We define $\gamma = arccos(\cos\theta + \lambda C_f \sin\theta^2)$ for $0 \leq \gamma \leq \pi$ under the requirement that

$$-\frac{1 + \cos\theta}{C_f \sin\theta^2} \leq \lambda \leq \frac{1 - \cos\theta}{C_f \sin\theta^2}.$$

Then we get that

$$\mathbb{E}_{\mathbf{w}}[e^{\lambda u}] = \frac{\sin\theta}{2\pi} \frac{\pi - \gamma}{\sin\gamma}.$$

Since $0 \leq \gamma, \theta \leq \pi$, now we further assume that $-\frac{b}{C \sin\theta^2} \leq \lambda \leq \frac{b}{C \sin\theta^2}$, then $\cos\theta - b \leq \cos\gamma \leq \cos\theta + b$. For a enough small b, we have that

$$\mathbb{E}_{\mathbf{w}}[e^{\lambda u}] \leq \frac{3(\pi - \theta)}{4\pi}. \tag{7}$$

From [7],

$$\mathbb{E}_{\mathbf{w}}[\lambda u] = \frac{2\lambda C_f}{\pi}(\sin\theta + \cos\theta(\pi - \theta)).$$

Therefore, we combine it with Eq. (7) to get that

$$\mathbb{E}_{\mathbf{w}}[e^{\lambda(u - \mathbb{E}_{\mathbf{w}}[u])}] \leq \frac{3(\pi - \theta)}{4\pi} \exp(-\lambda \frac{2C_f(\sin\theta + \cos\theta(\pi - \theta))}{\pi}),$$

for $-\frac{b}{C \sin\theta^2} \leq \lambda \leq \frac{b}{C \sin\theta^2}$ with a enough small b.

Because for $\frac{\pi}{2} \leq \theta \leq \pi$, $\cos\theta(\pi - \theta) \geq \cos\theta\tan(\pi - \theta) = -\sin\theta \geq 0$, we always can define $c = \frac{2C_f(\sin\theta + \cos\theta(\pi - \theta))}{\pi} \geq 0$ and it monotonically decreases to zero at $\theta = \pi$. Then we define $\nu^2 = \frac{c^2}{2\log(\frac{4\pi}{3(\pi - \theta)})}$ that can guarantee

$$\mathbb{E}[e^{\lambda(u-\mathbb{E}[u])}] \leq \exp(\frac{\nu^2\lambda^2}{2}),$$

for $|\lambda| \leq \frac{b}{C\sin\theta^2}$. This means that u follows a sub-exponential distribution and now we can use concentration bound to derive that

$$\mathbb{P}[|u-\mathbb{E}[u]| \geq \epsilon] \leq \left\{ \begin{array}{l} 2e^{-\frac{\epsilon^2}{2\nu^2}}\ if\ 0 \leq \epsilon \leq \frac{\nu^2 b}{C\sin\theta^2} \\ 2e^{-\frac{\epsilon b}{2C\sin\theta^2}}\ for\ \epsilon > \frac{\nu^2 b}{C\sin\theta^2} \end{array} \right. .$$

Therefore, for a small error ϵ, we can consider that

$$\mathbb{P}[|u-\mathbb{E}[u]| \geq \epsilon] \leq 2e^{-\frac{\epsilon^2}{2\nu^2}}.$$

The concentration inequality also applied to the average of r independent random variable u_i, which are defined for r independent $\mathbf{w}_i$. It shows that

$$\mathbb{P}[|\frac{1}{r}\sum_{i=1}^{r} u_i - \mathbb{E}[u]| \geq \epsilon] \leq 2e^{-\frac{r\epsilon^2}{2\nu^2}}. \tag{8}$$

Therefore, we obtain the concentration bound for fixed $\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))$ and $\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'))$.
Next, we show the result for a set $\mathcal{M}$. When $||\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))||_2||\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'))||_2$ is bounded and the angle between $\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))$ and $\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'))$ lies in $(0,\pi)$ meaning there is no collinearity, then we have an upper bound $\nu_{\mathcal{M}}$ for ν depending on that two conditions. Therefore, we similar choose T balls with radius δ to cover $\mathcal{M}$ as in the proof for Theorem 1. Let $\{\Delta_i\}_{i=1}^{T}$ denote the centers of these balls. Then using (8) and union bounds, for any two centers, such as Δ_1 and Δ_2, with probability at least $1-2\exp\left(\log(T^2) - \frac{r\epsilon^2}{8\nu_{\mathcal{M}}^2}\right)$,

$$|\hat{\mathcal{K}}^{\ell}(\Delta_1,\Delta_2) - \mathcal{K}^{\ell}(\Delta_1,\Delta_2)| \leq \frac{\epsilon}{2}. \tag{9}$$

Then similarly as in the proof for Theorem 0, for the function $\mathbf{u}(\mathbf{x},\mathbf{x}') = \hat{\mathcal{K}}^{\ell}(\mathbf{f}^{\ell-1}(\mathbf{x}),\mathbf{f}^{\ell-1}(\mathbf{x}')) - \mathcal{K}^{\ell}(\mathbf{f}^{\ell-1}(\mathbf{x}),\mathbf{f}^{\ell-1}(\mathbf{x}'))$, we have the inequality from partial derivative that

$$|\mathbf{u}(\mathbf{x},\mathbf{x}') - \mathbf{u}(\mathbf{x}_0,\mathbf{x}_0')| \leq L_{\sigma_{\mathcal{K}}}(||\sigma(\mathbf{f}^{\ell-1}(\mathbf{x})) - \sigma(\mathbf{f}^{\ell-1}(\mathbf{x}_0))||_2 + ||\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}')) - \sigma(\mathbf{f}^{\ell-1}(\mathbf{x}_0'))||_2), \tag{10}$$

where

$$
\begin{aligned}
L_{\sigma_{\mathcal{K}}} &= \arg \max_{\sigma(\mathbf{f}^{\ell-1}(\mathbf{x})),\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'))\in\mathcal{M}} ||\frac{1}{r}\sum_{i=1}^{r}\frac{\partial\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))^T\mathbf{w}_i)}{\partial\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))}\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'))^T\mathbf{w}_i) \\
&- \mathbb{E}_{\mathbf{w}}\frac{\partial\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))^T\mathbf{w})}{\partial\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))}\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'))^T\mathbf{w})||_2 \\
&= ||\frac{1}{r}\sum_{i=1}^{r}\frac{\partial\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}^*))^T\mathbf{w}_i)}{\partial\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))}\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'^*))^T\mathbf{w}_i) \\
&- \mathbb{E}_{\mathbf{w}}\frac{\partial\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}^*))^T\mathbf{w})}{\partial\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))}\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'^*))^T\mathbf{w})||_2.
\end{aligned}
$$

We also have that

$$
\begin{aligned}
\mathbb{E}L^2_{\sigma_{\mathcal{K}}} &= \frac{1}{r^2}\sum_{i=1}^{r}\mathbb{E}||\frac{\partial\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}^*))^T\mathbf{w}_i)}{\partial\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))}\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'^*))^T\mathbf{w}_i)||_2^2 \\
&- \frac{1}{r^2}||\mathbb{E}_{\mathbf{w}}\frac{\partial\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}^*))^T\mathbf{w})}{\partial\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))}\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'^*))^T\mathbf{w})||_2^2 \\
&\leq \frac{1}{r^2}\sum_{i=1}^{r}\mathbb{E}||\frac{\partial\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}^*))^T\mathbf{w}_i)}{\partial\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}))}\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'^*))^T\mathbf{w}_i)||_2^2.
\end{aligned}
$$

Since $\sigma_{\mathcal{K}}$ is ReLU, $||\sigma'_{\mathcal{K}}||_\infty \leq 1$, therefore we get that

$$
\mathbb{E}L^2_{\sigma_{\mathcal{K}}} \leq \frac{1}{r}\mathbb{E}|\sigma_{\mathcal{K}}(\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'^*))^T\mathbf{w})|||\mathbf{w}||_2^2.
$$

Again, since $\mathbf{w}$ follows a normal distribution that is symmetric, we choose axis to satisfy $\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'^*)) = \mathbf{e}_1||\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'^*))||_2$. Then we do a calculation,

$$
\begin{aligned}
\mathbb{E}L^2_{\sigma_{\mathcal{K}}} &\leq \frac{1}{r}||\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'^*))||_2\int_0^\infty dw_1\left\{\int_{-\infty}^{\infty}dw_2...\int_{-\infty}^{\infty}dw_d(w_1\sum_{j=1}^{d}w_j^2)p(w_2,...,w_d)\right\}p(w_1) \\
&= \frac{1}{r}||\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'^*))||_2\int_0^\infty dw_1\{w_1^3+(d-1)w_1)p(w_1)\} \\
&= \frac{1}{r}||\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'^*))||_2\int_0^\infty dw_1\{(d+3)w_1p(w_1)\} \\
&= \frac{1}{r}||\sigma(\mathbf{f}^{\ell-1}(\mathbf{x}'^*))||_2\frac{\sqrt{2}(d+3)}{\sqrt{\pi}} \\
&\leq \frac{\sqrt{2}(d+3)c_{\mathcal{M}}}{\sqrt{\pi}r}
\end{aligned}
$$

Therefore, by Markov's inequality,

$$\mathbb{P}(L_{\sigma_{\mathcal{K}}} \geq \frac{\epsilon}{4\delta}) \leq \mathbb{E}L^2_{\sigma_{\mathcal{K}}}\frac{16\delta^2}{\epsilon^2} \leq \frac{16\sqrt{2}(d+3)c_{\mathcal{M}}\delta^2}{\sqrt{\pi}\epsilon^2 r}.$$

Then using Eq. (10), with probability at least $1 - \frac{16\sqrt{2}(d+3)c_{\mathcal{M}}\delta^2}{\sqrt{\pi}\epsilon^2 r}$,

$$|u(\mathbf{x}, \mathbf{x}') - u(\mathbf{x}_0, \mathbf{x}'_0)| \leq \frac{\epsilon}{2}.$$

This inequality combined with Eq. (9) enables us to conclude that

$$\sup_{\sigma(\mathbf{f}^{\ell-1}(\mathbf{x})), \sigma(\mathbf{f}^{\ell-1}(\mathbf{x}')) \in \mathcal{M}} |\hat{\mathcal{K}}^\ell(\mathbf{f}^{\ell-1}(\mathbf{x}), \mathbf{f}^{\ell-1}(\mathbf{x}')) - \mathcal{K}^\ell(\mathbf{f}^{\ell-1}(\mathbf{x}), \mathbf{f}^{\ell-1}(\mathbf{x}'))| \leq \epsilon.$$

with probability at least $1 - \frac{16\sqrt{2}(d+3)c_{\mathcal{M}}\delta^2}{\sqrt{\pi}\epsilon^2 r} - 2\exp\left(\log(T^2) - \frac{r\epsilon^2}{8\nu^2_{\mathcal{M}}}\right)$. Recall that $T = (4diam(\mathcal{M})/\delta)^d$, so the probability has a format of $1 - \kappa_1\delta^2 - \kappa_2\delta^{-2d}$ for δ. By setting $\delta = \frac{\kappa_2}{\kappa_1}^{\frac{1}{2+2d}}$, we have the probability as $1 - 2\kappa_1^{\frac{2d}{2+2d}}\kappa_2^{\frac{2}{2+2d}}$. So the probability is at least

$$1 - 2^{10}\left(\frac{c_{\mathcal{M}}(d+3)diam(\mathcal{M})^2}{\epsilon^2 r}\right)^{\frac{d}{1+d}} \exp\left\{-\frac{r\epsilon^2}{8(1+d)\nu^2_{\mathcal{M}}}\right\}.$$

□

6.2.2 The relation between random feature [8] and inducing points approximation [22]

First, we review the algorithm in [22] based on inducing points and doubly stochastic variational inference. In the background section, we introduced that a L layer DGP can be represented by

$$p(\mathbf{y}, \{\mathbf{F}^\ell\}_{\ell=1}^L) = \prod_{i=1}^n p(y_i|f_i^L)\prod_{\ell=1}^L p(\mathbf{F}^\ell|\mathbf{F}^{\ell-1}),$$

where $\mathbf{F}^\ell \in \mathbb{R}^{n\times d}$ for $0 \leq \ell < L$ and $\mathbf{F}^L \in \mathbb{R}^{n\times 1}$, with $\mathbf{F}^\ell|\mathbf{F}^{\ell-1} \sim N(0, \mathcal{K}^\ell(\mathbf{F}^{\ell-1}, \mathbf{F}^{\ell-1}))$.
In [22], they further define an additional set of m inducing points $\mathbf{Z}^\ell = (\mathbf{z}_1^\ell, ..., \mathbf{z}_m^\ell)$ for each layer $0 \leq \ell < L$. We use the notation $\mathbf{u}^\ell = f^\ell(\mathbf{Z}^{\ell-1})$ for the function values at the inducing points. Since we have d output on layer ℓ, we use $\mathbf{U}^\ell \in \mathbb{R}^{m\times d}$ for the function value matrix at the inducing points. By the definition of GP, the joint density $p(\mathbf{F}^\ell, \mathbf{U}^\ell)$ is a Gaussian distribution given inputs from previous layer. Therefore, we have the joint posterior of $\mathbf{y}, \{\mathbf{F}^\ell, \mathbf{U}^\ell\}_{\ell=1}^L$ is

$$p(\mathbf{y}, \{\mathbf{F}^\ell, \mathbf{U}^\ell\}_{\ell=1}^L) = \prod_{i=1}^n p(y_i|f_i^L)\prod_{\ell=1}^L p(\mathbf{F}^\ell|\mathbf{U}^\ell; \mathbf{F}^{\ell-1}, \mathbf{Z}^{\ell-1})p(\mathbf{U}^\ell; \mathbf{Z}^{\ell-1}).$$

The posterior of $\{\mathbf{F}^\ell, \mathbf{U}^\ell\}_{\ell=1}^L$ is intractable, so the authors in [22] define the variational posterior

$$q(\{\mathbf{F}^\ell, \mathbf{U}^\ell\}_{\ell=1}^L) = \prod_{\ell=1}^L p(\mathbf{F}^\ell | \mathbf{U}^\ell; \mathbf{F}^{\ell-1}, \mathbf{Z}^{\ell-1}) q(\mathbf{U}^\ell),$$

with $q(\mathbf{U}^\ell) = \prod_{j=1}^d q(\mathbf{U}_{.j}^\ell)$ and $q(\mathbf{U}_{.j}^\ell) \sim N(\mathbf{m}_j^\ell, \mathbf{S}_j^\ell)$. Then they calculate the evidence lower bound of the DGP, which is

$$\text{ELBO}_{DGP} = \mathbb{E}_{q(\{\mathbf{F}^\ell, \mathbf{U}^\ell\}_{\ell=1}^L)} \left[\frac{p(\mathbf{y}, \{\mathbf{F}^\ell, \mathbf{U}^\ell\}_{\ell=1}^L}{q(\{\mathbf{F}^\ell, \mathbf{U}^\ell\}_{\ell=1}^L)} \right].$$

Based on the definition of $q(\{\mathbf{F}^\ell, \mathbf{U}^\ell\}_{\ell=1}^L)$, we can simplify ELBO_{DGP} and show that it is equal as

$$\text{ELBO}_{DGP} = \sum_{i=1}^n \mathbb{E}_{q(f_i^L)}[\log p(y_i | f_i^L)] - \sum_{\ell=1}^L KL[q(\mathbf{U}^\ell) | p(\mathbf{U}^\ell; \mathbf{Z}^{\ell-1})]. \tag{11}$$

From [22], after marginalizing the inducing variables from each layer analytically, we can show that

$$q(\{\mathbf{F}^\ell\}_{\ell=1}^L) = \prod_{\ell=1}^L q(\mathbf{F}^\ell | \mathbf{m}^\ell, \mathbf{S}^\ell; \mathbf{F}^{\ell-1}, \mathbf{Z}^{\ell-1}) = \prod_{\ell=1}^L N(\mathbf{F}^\ell | \boldsymbol{\mu}^\ell, \boldsymbol{\Sigma}^\ell) = \prod_{\ell=1}^L \prod_{j=1}^d N(\mathbf{F}_{.j}^\ell | \tilde{\boldsymbol{\mu}}_j^\ell, \tilde{\boldsymbol{\Sigma}}_j^\ell). \tag{12}$$

Here,

$$\tilde{\boldsymbol{\mu}}_j^\ell = \alpha(\mathbf{F}^{\ell-1})^T \mathbf{m}_{.j}^\ell$$

$$\tilde{\boldsymbol{\Sigma}}_j^\ell = \mathcal{K}^\ell(\mathbf{F}^{\ell-1}, \mathbf{F}^{\ell-1}) - \alpha(\mathbf{F}^{\ell-1})^T (\mathcal{K}^\ell(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1}) - \mathbf{S}_{.j}) \alpha(\mathbf{F}^{\ell-1})$$

with $\alpha(\mathbf{F}^{\ell-1}) = \mathcal{K}^\ell(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1})^{-1} \mathcal{K}^\ell(\mathbf{Z}^{\ell-1}, \mathbf{F}^{\ell-1})$.
From Eq. (11), we only need to get $q(f_i^L)$ from $q(\{\mathbf{F}^\ell\}_{\ell=1}^L)$ for sample i with $1 \le i \le n$. In [22], they point out that based on the format of Eq. (12),

$$q(f_i^L) = \int \cdots \int \prod_{\ell=1}^L q(\mathbf{F}_{i.}^\ell | \mathbf{m}_{i.}^\ell, \mathbf{S}_{i,}^\ell; \mathbf{F}_{i.}^{\ell-1}, \mathbf{Z}^{\ell-1}) d\mathbf{F}_{i.}^{\ell-1},$$

which means that the ith marginal of the final layer of the variational DGP for sample i depends only on the ith marginals of all the other layers.

Proof of Theorem 2

Theorem 2. Using the variational approximation [22] for the posterior of a DGP defined on $\{\hat{\mathcal{K}}^\ell\}_{\ell=1}^L$ with inducing points, we obtain exactly the same variational posterior $q(\{\mathbf{F}^\ell\}_{\ell=1}^L)$ and evidence lower bound ELBO as the variational posterior for $\mathcal{N}(\mathcal{S})$.

Proof. To show the equivalence of evidence lower bound, we only need to guarantee that $q(\mathbf{F}^{\ell}_{i.}|\mathbf{m}^{\ell}_{i.}, \mathbf{S}^{\ell}_{i,}; \mathbf{F}^{\ell-1}_{i.}, \mathbf{Z}^{\ell-1})$ and $KL[q(\mathbf{U}^{\ell})|p(\mathbf{U}^{\ell}; \mathbf{Z}^{\ell-1})]$ are the same as the relevant values for $\mathcal{N}(\mathcal{S})$ for all $1 \leq i \leq n$ and $1 \leq \ell \leq L$. We also need to show the equivalence between variational posterior $q(\{\mathbf{F}^{\ell}\}_{\ell=1}^{L})$ for the two methods. All those can be satisfied by showing the equivalence that $q(\mathbf{F}^{\ell}|\mathbf{m}^{\ell}, \mathbf{S}^{\ell}; \mathbf{F}^{\ell-1}, \mathbf{Z}^{\ell-1})$ and $KL[q(\mathbf{U}^{\ell})|p(\mathbf{U}^{\ell}; \mathbf{Z}^{\ell-1})]$ are the same as the relevant values for $\mathcal{N}(\mathcal{S})$ for all $1 \leq \ell \leq L$. For both two methods, since for each layer ℓ, the d outputs are independent, so the posterior distribution can be decomposed into a product of d terms and the KL divergence can be decomposed into a summation of d terms. We only need to prove the result for a single j with $1 \leq j \leq d$ and a single ℓ with $1 \leq \ell \leq L$.
Based on Eq. (12), for $q(\mathbf{F}^{\ell}_{.j}|\mathbf{m}^{\ell}_{.j}, \mathbf{S}^{\ell}_{.j}; \mathbf{F}^{\ell-1}, \mathbf{Z}^{\ell-1})$, the mean and variances are

$$\tilde{\boldsymbol{\mu}}^{\ell}_{j} = \alpha(\mathbf{F}^{\ell-1})^{T}\mathbf{m}^{\ell}_{.j}$$

$$\tilde{\boldsymbol{\Sigma}}^{\ell}_{j} = \hat{\mathcal{K}}^{\ell}(\mathbf{F}^{\ell-1}, \mathbf{F}^{\ell-1}) - \alpha(\mathbf{F}^{\ell-1})^{T}(\hat{\mathcal{K}}^{\ell}(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1}) - \mathbf{S}_{.j})\alpha(\mathbf{F}^{\ell-1})$$

with $\alpha(\mathbf{F}^{\ell-1}) = \hat{\mathcal{K}}^{\ell}(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1})^{-1}\hat{\mathcal{K}}^{\ell}(\mathbf{Z}^{\ell-1}, \mathbf{F}^{\ell-1})$. We can decompose the kernel into $\hat{\mathcal{K}}^{\ell}(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1}) = \Phi^{\ell}(\mathbf{Z}^{\ell-1})\Phi^{\ell}(\mathbf{Z}^{\ell-1})^{T}$. Now we choose the number of inducing points m as $m = r$, then we have a square matrix $\Phi^{\ell}(\mathbf{Z}^{\ell-1})$ with each entry is independently identically from a distribution based on the random feature weight vector $\mathbf{w}_{j}$ and the random inducing points $\mathbf{Z}^{\ell-1}_{i.}$.
For continuous $\sigma_{\mathcal{K}}$ and σ, every entry in $\Phi^{\ell}(\mathbf{Z}^{\ell-1})$ is absolutely continuous with respect to Lebesgue measure since we can define density function. Then based on random matrix theory [20, 25], the square matrix $\Phi^{\ell}(\mathbf{Z}^{\ell-1})$ is almost surely invertible. Therefore, we treat $\Phi^{\ell}(\mathbf{Z}^{\ell-1})$ as an invertible matrix in following analysis.
Replace $\hat{\mathcal{K}}^{\ell}(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1})$ by $\Phi^{\ell}(\mathbf{Z}^{\ell-1})\Phi^{\ell}(\mathbf{Z}^{\ell-1})^{T}$, we have that

$$\tilde{\boldsymbol{\mu}}^{\ell}_{j} = \Phi^{\ell}(\mathbf{F}^{\ell-1})(\Phi^{\ell}(\mathbf{Z}^{\ell-1})^{T}\Phi^{\ell}(\mathbf{Z}^{\ell-1}))^{-1}\Phi^{\ell}(\mathbf{Z}^{\ell-1})^{T}\mathbf{m}^{\ell}_{.j}$$

$$\tilde{\boldsymbol{\Sigma}}^{\ell}_{j} = \Phi^{\ell}(\mathbf{F}^{\ell-1})\Phi^{\ell}(\mathbf{F}^{\ell-1})^{T} - \alpha(\mathbf{F}^{\ell-1})^{T}(\Phi^{\ell}(\mathbf{Z}^{\ell-1})\Phi^{\ell}(\mathbf{Z}^{\ell-1})^{T} - \mathbf{S}_{.j})\alpha(\mathbf{F}^{\ell-1})$$

Through simple algebra, we get that

$$\begin{aligned} \tilde{\boldsymbol{\mu}}^{\ell}_{j} &= \alpha(\mathbf{F}^{\ell-1})^{T}\mathbf{m}^{\ell}_{.j} \\ \tilde{\boldsymbol{\Sigma}}^{\ell}_{j} &= \alpha(\mathbf{F}^{\ell-1})^{T}\mathbf{S}_{.j}\alpha(\mathbf{F}^{\ell-1}) \end{aligned} \tag{13}$$

Since $\Phi^{\ell}(\mathbf{Z}^{\ell-1})$ is invertible, we define

$$\mathbf{m}^{\ell}_{.j} = \Phi^{\ell}(\mathbf{Z}^{\ell-1})\boldsymbol{\mu}^{\ell}_{j,new}$$

$$\mathbf{S}_{.j} = \Phi^{\ell}(\mathbf{Z}^{\ell-1})\boldsymbol{\Sigma}^{\ell}_{j,new}\Phi^{\ell}(\mathbf{Z}^{\ell-1})^{T},$$

and plug them into Eq. (13) then we get

$$
\begin{aligned}
\tilde{\boldsymbol{\mu}}_j^\ell &= \Phi^\ell(\mathbf{F}^{\ell-1})\boldsymbol{\mu}_{j,new}^\ell \\
\tilde{\boldsymbol{\Sigma}}_j^\ell &= \Phi^\ell(\mathbf{F}^{\ell-1})\boldsymbol{\Sigma}_{j,new}^\ell \Phi^\ell(\mathbf{F}^{\ell-1})^T.
\end{aligned}
\tag{14}
$$

In our BNN construction for $\mathcal{N}(\mathcal{S})$, the variational posterior over $\mathbf{V}$ leads to $\mathbf{F}_{\cdot j}^\ell = \Phi^\ell(\mathbf{F}^{\ell-1})\mathbf{v}_j^\ell$ with $\mathbf{v}_j^\ell \sim N(\boldsymbol{\mu}_{j,new}^\ell, \boldsymbol{\Sigma}_{j,new}^\ell)$. Then we have that

$$\mathbf{F}_{\cdot j}^\ell \sim N(\Phi^\ell(\mathbf{F}^{\ell-1})\boldsymbol{\mu}_{j,new}^\ell, \Phi^\ell(\mathbf{F}^{\ell-1})\boldsymbol{\Sigma}_{j,new}^\ell \Phi^\ell(\mathbf{F}^{\ell-1})^T).$$

That is identical with the results from inducing points method in Eq. (14). Based on this construction, we also have that

$$\mathbf{U}_{\cdot j}^\ell = \Phi^\ell(\mathbf{Z}^{\ell-1})\mathbf{v}_j^\ell$$

Since the KL divergence is invariant under parameter transformations, we have that

$$KL(q(\mathbf{U}_{\cdot j}^\ell)||p(\mathbf{U}_{\cdot j}^\ell)) = KL(q(\mathbf{v}_j^\ell)||p(\mathbf{v}_j^\ell))$$

□

Proof of Theorem 3

For a kernel $\mathcal{K}^\ell$ belongs to a general class, we can still use a similar technique as in the proof of Theorem 2 to show the equivalence. However, this time we cannot use the random feature matrix $\Phi^\ell(\mathbf{F}^{\ell-1})\Phi^\ell(\mathbf{F}^{\ell-1})^T$ to approximate $\mathcal{K}^\ell(\mathbf{F}^{\ell-1}, \mathbf{F}^{\ell-1})$. It turns out that a good replacement for $\Phi^\ell(\mathbf{F}^{\ell-1})$ to approximate the basis of $\mathcal{K}^\ell(\mathbf{F}^{\ell-1}, \mathbf{F}^{\ell-1})$ is $\mathcal{K}^\ell(\mathbf{F}^{\ell-1}, \mathbf{Z}^{\ell-1})\mathcal{K}^\ell(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1})^{-1/2}$ which we will show shortly. The proof technique for Theorem 3 is similar as the technique for Theorem 2. However, for a general class of $\mathcal{K}^\ell$, $\mathcal{K}^\ell(\mathbf{F}^{\ell-1}, \mathbf{F}^{\ell-1})$ can be full rank which is equal to sample size n. Therefore, the difference from the approximation using the rank r basis $\mathcal{K}^\ell(\mathbf{F}^{\ell-1}, \mathbf{Z}^{\ell-1})\mathcal{K}^\ell(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1})^{-1/2}$ is $\mathcal{K}^\ell(\mathbf{F}^{\ell-1}, \mathbf{F}^{\ell-1}) - \mathcal{K}^\ell(\mathbf{F}^{\ell-1}, \mathbf{Z}^{\ell-1})\mathcal{K}^\ell(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1})^{-1}\mathcal{K}^\ell(\mathbf{Z}^{\ell-1}, \mathbf{F}^{\ell-1})$. This is the constant offset that does not depend on training which we mention in Theorem 3. For the optimization of ELBO, only the diagonal terms in this offset matrix is used so we can also add this into BNN as a bias term with random weight that we do not train.

Remark. After the optimization of ELBO, one can get the uncertainty estimates from the variational posterior. If $\mathcal{K}^\ell(\mathbf{F}^{\ell-1}, \mathbf{F}^{\ell-1}) - \mathcal{K}^\ell(\mathbf{F}^{\ell-1}, \mathbf{Z}^{\ell-1})\mathcal{K}^\ell(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1})^{-1}\mathcal{K}^\ell(\mathbf{Z}^{\ell-1}, \mathbf{F}^{\ell-1})$ is not present, then one can choose $\mathbf{V}$ from its variational posterior and the output estimates for every samples directly come from one pass of feed-forward neural network. However, when $\mathcal{K}^\ell(\mathbf{F}^{\ell-1}, \mathbf{F}^{\ell-1}) - \mathcal{K}^\ell(\mathbf{F}^{\ell-1}, \mathbf{Z}^{\ell-1})\mathcal{K}^\ell(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1})^{-1}\mathcal{K}^\ell(\mathbf{Z}^{\ell-1}, \mathbf{F}^{\ell-1})$ exists, n passes of feed-forward neural network computation for n samples need to depend on each other to derive the outputs. In [22], they also permit the prior of DGP to have a non-zero mean function, which can lead to another offset if the prior mean of DGP at each layer is non-zero. Similarly, this offset does not depend on training and can be included into BNN as a bias term with random weights that we do not train.

Another term that is usually discussed in DGP is the noisy corruption. In this result for general kernel $\mathcal{K}$ in Thoerem 3, in [22], the authors show that the noisy corruption can be included into the kernel $\mathcal{K}$. For our earlier result in Theorem 2, we do not include the noisy corruption in intermediate layers, since the complexity of intermediate function is already restricted by the rank r. It does not overfit the data so we do not need the noisy corruption which is usually used to avoid overfitting when the kernel basis has infinite dimension which can be super expressive.
Now we review the definition of IPB and Theorem 3, then we present the proof for Theorem 3.

Definition 1. For a kernel $\mathcal{K}$, IPB can be constructed by choosing r additional points $\mathbf{Z}$ (inducing points), taking the inputs $\mathbf{x}$ and outputting an r-dimension vector $\mathcal{K}(\mathbf{x}, \mathbf{Z})\mathcal{K}(\mathbf{Z}, \mathbf{Z})^{-1/2}$.

Theorem 3. Using the variational approximation [22] for the posterior of a DGP defined on $\{\mathcal{K}^\ell\}_{\ell=1}^L$ with inducing points, we can obtain the same variational posterior $q(\{\mathbf{F}^\ell\}_{\ell=1}^L)$ and evidence lower bound ELBO as the variational posterior for $\mathcal{N}(\mathcal{S})$ (with IPB) except a constant offset that does not depend on training.

Proof. To show the equivalence of evidence lower bound, we only need to guarantee that $q(\mathbf{F}^\ell_{i.}|\mathbf{m}^\ell_{i.}, \mathbf{S}^\ell_{i,}; \mathbf{F}^{\ell-1}_{i.}, \mathbf{Z}^{\ell-1})$ and $KL[q(\mathbf{U}^\ell)|p(\mathbf{U}^\ell; \mathbf{Z}^{\ell-1})]$ are the same as the relevant values for $\mathcal{N}(\mathcal{S})$ for all $1 \le i \le n$ and $1 \le \ell \le L$. We also need to show the equivalence between variational posterior $q(\{\mathbf{F}^\ell\}_{\ell=1}^L)$ for the two methods. All those can be satisfied by showing the equivalence that $q(\mathbf{F}^\ell|\mathbf{m}^\ell, \mathbf{S}^\ell; \mathbf{F}^{\ell-1}, \mathbf{Z}^{\ell-1})$ and $KL[q(\mathbf{U}^\ell)|p(\mathbf{U}^\ell; \mathbf{Z}^{\ell-1})]$ are the same as the relevant values for $\mathcal{N}(\mathcal{S})$ for all $1 \le \ell \le L$. For both two methods, since for each layer ℓ, the d outputs are independent, so the posterior distribution can be decomposed into a product of d terms and the KL divergence can be decomposed into a summation of d terms. We only need to prove the result for a single j with $1 \le j \le d$ and a single ℓ with $1 \le \ell \le L$.
Based on Eq. (12), for $q(\mathbf{F}^\ell_{.j}|\mathbf{m}^\ell_{.j}, \mathbf{S}^\ell_{.j}; \mathbf{F}^{\ell-1}, \mathbf{Z}^{\ell-1})$, the mean and variances are

$$\tilde{\boldsymbol{\mu}}^\ell_j = \alpha(\mathbf{F}^{\ell-1})^T \mathbf{m}^\ell_{.j}$$

$$\tilde{\boldsymbol{\Sigma}}^\ell_j = \mathcal{K}^\ell(\mathbf{F}^{\ell-1}, \mathbf{F}^{\ell-1}) - \alpha(\mathbf{F}^{\ell-1})^T(\mathcal{K}^\ell(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1}) - \mathbf{S}_{.j})\alpha(\mathbf{F}^{\ell-1})$$

with $\alpha(\mathbf{F}^{\ell-1}) = \mathcal{K}^\ell(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1})^{-1}\mathcal{K}^\ell(\mathbf{Z}^{\ell-1}, \mathbf{F}^{\ell-1})$.
We use r to refer the number of inducing points and **we denote the IPB block matrix** $\mathcal{K}^\ell(\mathbf{F}^{\ell-1}, \mathbf{Z}^{\ell-1})\mathcal{K}^\ell(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1})^{-1/2}$ as $\mathrm{IP}^\ell(\mathbf{F}^{\ell-1})$, then we notice that

$$\begin{aligned}
\tilde{\boldsymbol{\mu}}^\ell_j &= \mathrm{IP}^\ell(\mathbf{F}^{\ell-1})\mathcal{K}^\ell(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1})^{-1/2}\mathbf{m}^\ell_{.j} \\
\tilde{\boldsymbol{\Sigma}}^\ell_j &= \mathrm{IP}^\ell(\mathbf{F}^{\ell-1})\mathcal{K}^\ell(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1})^{-1/2}\mathbf{S}_{.j}\mathcal{K}^\ell(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1})^{-1/2}\mathrm{IP}^\ell(\mathbf{F}^{\ell-1})^T \\
&\quad + \mathcal{K}^\ell(\mathbf{F}^{\ell-1}, \mathbf{F}^{\ell-1}) - \mathcal{K}^\ell(\mathbf{F}^{\ell-1}, \mathbf{Z}^{\ell-1})\mathcal{K}^\ell(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1})^{-1}\mathcal{K}^\ell(\mathbf{Z}^{\ell-1}, \mathbf{F}^{\ell-1}).
\end{aligned} \tag{15}$$

We have discussed the second term in $\tilde{\boldsymbol{\Sigma}}^\ell_j$ which is a constant offset that does not depend on training. Therefore we assume it to be zero in following analysis then we get the exactly same result as our variational posterior approximation for $\mathcal{N}(\mathcal{S})$ when IPB is used.

We define

$$\mathbf{m}^{\ell}_{\cdot j} = \mathcal{K}^{\ell}(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1})^{1/2}\boldsymbol{\mu}^{\ell}_{j,new}$$

$$\mathbf{S}_{\cdot j} = \mathcal{K}^{\ell}(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1})^{1/2}\boldsymbol{\Sigma}^{\ell}_{j,new}\mathcal{K}^{\ell}(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1})^{1/2},$$

and plug them into Eq. (15) then we get

$$\begin{aligned}\tilde{\boldsymbol{\mu}}^{\ell}_{j} &= \mathrm{IP}^{\ell}(\mathbf{F}^{\ell-1})\boldsymbol{\mu}^{\ell}_{j,new} \\ \tilde{\boldsymbol{\Sigma}}^{\ell}_{j} &= \mathrm{IP}^{\ell}(\mathbf{F}^{\ell-1})\boldsymbol{\Sigma}^{\ell}_{j,new}\mathrm{IP}^{\ell}(\mathbf{F}^{\ell-1})^{T}.\end{aligned} \tag{16}$$

In our BNN construction for $\mathcal{N}(\mathcal{S})$, the variational posterior over $\mathbf{V}$ leads to $\mathbf{F}^{\ell}_{\cdot j} = \mathrm{IP}^{\ell}(\mathbf{F}^{\ell-1})\mathbf{v}^{\ell}_{j}$ with $\mathbf{v}^{\ell}_{j} \sim N(\boldsymbol{\mu}^{\ell}_{j,new}, \boldsymbol{\Sigma}^{\ell}_{j,new})$. Then we have that

$$\mathbf{F}^{\ell}_{\cdot j} \sim N(\mathrm{IP}^{\ell}(\mathbf{F}^{\ell-1})\boldsymbol{\mu}^{\ell}_{j,new}, \mathrm{IP}^{\ell}(\mathbf{F}^{\ell-1})\boldsymbol{\Sigma}^{\ell}_{j,new}\mathrm{IP}^{\ell}(\mathbf{F}^{\ell-1})^{T}).$$

That is identical with the results from inducing points method in Eq. (16). Based on this construction, we also have that

$$\mathbf{U}^{\ell}_{\cdot j} = \mathrm{IP}^{\ell}(\mathbf{Z}^{\ell-1})\mathbf{v}^{\ell}_{j} = \mathcal{K}^{\ell}(\mathbf{Z}^{\ell-1}, \mathbf{Z}^{\ell-1})^{1/2}\mathbf{v}^{\ell}_{j}$$

Since the KL divergence is invariant under parameter transformations, we have that

$$KL(q(\mathbf{U}^{\ell}_{\cdot j})||p(\mathbf{U}^{\ell}_{\cdot j})) = KL(q(\mathbf{v}^{\ell}_{j})||p(\mathbf{v}^{\ell}_{j}))$$

□

6.2.3 Proof of Theorem 4

The post-training ANOVA decomposition is

$$\mathrm{I}^{n}_{T}(\mathbf{x}_{T}) = \prod_{i\in T}(I_{x_i} - \mathbb{E}^{n}_{x_i})\prod_{j\notin T}\mathbb{E}^{n}_{x_j}f(x_1, ..., x_p), \tag{17}$$

Theorem 4. If there exist inputs clusters $\{T^*_j\}_{j=1}^{k^*}$ such that $f^*(\mathbf{x}) = \sum_{j=1}^{k^*} g^*_j(\mathbf{x}_{T^*_j})$ with k^* at the order of polynomial in p and $c = \max_{j=1}^{k^*}|T^*_j| = O(\log p)$, then there exists a trained AddNN that predicts $\mathbf{y}$ well and restricts the number of possible interactions at polynomial in p. Further, if every sub neural network has L layers with d hidden units, then the computation complexity of measure (17) is at most $n^c k^* d^{2L-1}$, which is also polynomial in p.

Proof. In that case, there exists a trained AddNN f which is $\hat{f}(\mathbf{x}) = \sum_{j=1}^{k^*} \hat{g}_j(\mathbf{x}_{T_j^*})$ with the same k^* and inputs clusters $\{T_j^*\}_{j=1}^{k^*}$. For such $\hat{f}$, without knowing the truth, every possible interaction is a subset of one inputs cluster T_j^* for some j. Therefore, the number of possible interactions is bounded by $\sum_{j=1}^{k^*} 2^{|T_j^*|} \le k^* 2^c$, so it is polynomial in p. For an interaction among a subset S, we have that

$$\begin{aligned}
\mathrm{I}_S^n(\mathbf{x}_S) &= \prod_{i\in S}(I_{x_i} - \mathbb{E}_{x_i}^n)\prod_{j\notin S}\mathbb{E}_{x_j}^n \hat{f}(x_1,...,x_p)\\
&= \prod_{i\in S}(I_{x_i} - \mathbb{E}_{x_i}^n)\prod_{j\notin S}\mathbb{E}_{x_j}^n \sum_{m=1}^{k^*}\hat{g}_m(\mathbf{x}_{T_m^*})\\
&= \sum_{m=1}^{k^*}\prod_{i\in S}(I_{x_i} - \mathbb{E}_{x_i}^n)\prod_{j\notin S}\mathbb{E}_{x_j}^n \hat{g}_m(\mathbf{x}_{T_m^*})\\
&= \sum_{m:S\subseteq T_m^*}\prod_{i\in S}(I_{x_i} - \mathbb{E}_{x_i}^n)\prod_{j\notin S}\mathbb{E}_{x_j}^n \hat{g}_m(\mathbf{x}_{T_m^*}),
\end{aligned}$$

because for m that $S \not\subseteq T_m^*$, then there exists an $i_0 \in S$ such that $i_0 \notin T_m^*$, then

$$\prod_{i\in S}(I_{x_i} - \mathbb{E}_{x_i}^n)\prod_{j\notin S}\mathbb{E}_{x_j}^n \hat{g}_m(\mathbf{x}_{T_m^*}) = 0$$

because $(I_{x_{i_0}} - \mathbb{E}_{x_{i_0}}^n)\hat{g}_m(\mathbf{x}_{T_m^*}) = \hat{g}_m(\mathbf{x}_{T_m^*}) - \hat{g}_m(\mathbf{x}_{T_m^*}) = 0$. We can further simply $\mathrm{I}_S^n(\mathbf{x}_S)$,

$$\begin{aligned}
\mathrm{I}_S^n(\mathbf{x}_S) &= \sum_{m:S\subseteq T_m^*}\prod_{i\in S}(I_{x_i} - \mathbb{E}_{x_i}^n)\prod_{j\notin S}\mathbb{E}_{x_j}^n \hat{g}_m(\mathbf{x}_{T_m^*})\\
&= \sum_{m:S\subseteq T_m^*}\prod_{i\in S}(I_{x_i} - \mathbb{E}_{x_i}^n)\prod_{j\notin S, j\in T_m^*}\mathbb{E}_{x_j}^n \hat{g}_m(\mathbf{x}_{T_m^*}).
\end{aligned}$$

Therefore, the computation complexity of $\mathrm{I}_S^n(\mathbf{x}_S)$ is equal to the computation complexity for

$$\cup_{m:S\subseteq T_m^*}\left\{\prod_{i\in S}(I_{x_i} - \mathbb{E}_{x_i}^n)\prod_{j\notin S, j\in T_m^*}\mathbb{E}_{x_j}^n \hat{g}_m(\mathbf{x}_{T_m^*})\right\}, \tag{18}$$

which is the union of $|m : S \subseteq T_m^*|$ functions where each function involves the calculation of feed-forward neural network and empirical evaluation. Therefore, to compute all interactions, we need to compute the union of (18) for all possible interactions S. Because every possible interaction is a subset of one inputs cluster T_j^* for some j, we can exchange the order of unions and we get that the computation complexity for all interactions is equal as evaluating

$$\cup_m \cup_{S\subseteq T_m^*}\left\{\prod_{i\in S}(I_{x_i} - \mathbb{E}_{x_i}^n)\prod_{j\notin S, j\in T_m^*}\mathbb{E}_{x_j}^n \hat{g}_m(\mathbf{x}_{T_m^*})\right\}. \tag{19}$$

To compute $\cup_{S\subseteq T_m^*}\left\{\prod_{i\in S}(I_{x_i}-\mathbb{E}^n_{x_i})\prod_{j\notin S,j\in T_m^*}\mathbb{E}^n_{x_j}\hat{g}_m\left(\mathbf{x}_{T_m^*}\right)\right\}$ for some m, we only need to compute

$$M^n_{T_m^*}(\mathbf{x}_{T_m^*})=\prod_{i\in T_m^*}I_{x_i}\hat{g}\left(\mathbf{x}_{T_m^*}\right),$$

which involves all the evaluations of the feed-forward neural network. To evaluate $\prod_{i\in S}(I_{x_i}-\mathbb{E}^n_{x_i})\prod_{j\notin S,j\in T_m^*}\mathbb{E}^n_{x_j}\hat{g}_m\left(\mathbf{x}_{T_m^*}\right)$ for $S\subseteq T_m^*$, the computation only involves basic addition operation given $M^n_{T_m^*}$. Therefore, we show that the evaluation of all possible interactions has the same computation complexity as evaluating

$$\cup_m\{M^n_{T_m^*}(\mathbf{x}_{T_m^*})\}=\cup_m\{\prod_{i\in T_m^*}I_{x_i}\hat{g}\left(\mathbf{x}_{T_m^*}\right)\}. \tag{20}$$

For every member in the union regarding m, the evaluation complexity is $n^{|T_m^*|}d^{2L-1}\le n^c d^{2L-1}$. Therefore, the computation complexity regarding (17) for all possible interactions based on model $\hat{f}$ is bounded by $n^c k^* d^{2L-1}$, which is polynomial in p when k^* is polynomial in p and c is the order of $O(\log p)$. $\square$

Remark 1. When the truth function f^* is additive, the function class of AddNN includes a member that can compute interactions and outputs from the model efficiently while the function class of an arbitrary NN make it impossible to learn such a model and always involve computation that is exponential in p. In practice, we always use group Lasso type penalty on the first layer to encourage each sub neural network to depend on few inputs to approach the truth function f^*.

Remark 2. We can use the measure in [28] as well, which can be seen as choosing one sample baseline instead of the average baseline in (17). In other words, now we use operation $\delta_{x_i}(\mathbf{x}_i^0)$ to replace $\mathbb{E}^n_{x_i}$ in (17) based on one sample $\mathbf{x}^0$ for $1\le i\le n$. Then the computation complexity of measure (17) does not depend on n both for AddNN and NN. In that case, the number of possible interactions for AddNN is still at polynomial in p and the number is exponential in p for an arbitrary NN. Also, the computation complexity of measure (17) is k^*d^{2L-1} for AddNN and $(k^*d)^{2L-1}$ for an arbitrary NN with L layers and k^*d hidden units where the NN requires $k^{*2(L-1)}$ times more computation. The choice of $\mathbb{E}^n$ as baseline, compared to $\delta(\mathbf{x}^0)$, is better for comparison with the population and can lead to a useful measure $||I^n_T(\mathbf{x}_T)||_{2,n}$ (the empirical ℓ_2 norm of the interaction), which can be used to detect the interactions. We give an example of their difference for explanation in the decision making process. For one who is interested in making an investment with x_1 dollars, the $\mathbb{E}^n$ baseline informs that, based on this investment, how much more one can earn than the average of the money that people earn. On the other side, the $\delta(\mathbf{x}_0)$ baseline informs that, at the current investment with x_0 dollars, if all other factors do not change, how much more one can earn if he/she decides to invest x_1-x_0 more dollars.

6.3 Model details for experiments

We discuss more details about Bayesian additive Neural Network (AddNN) implementation and provide more experimental results. In our experiments, we use 10 small(sub) neural networks, where each has 2

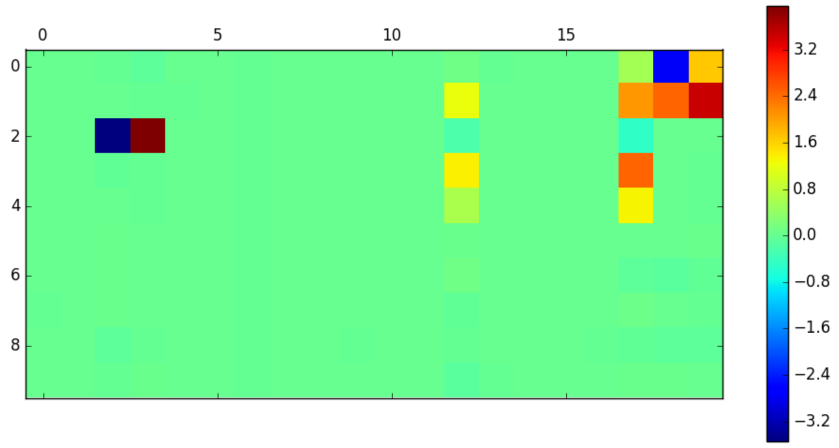


Figure 3: AddNN learns the additive structure of the function f_1. The rows of visualized matrix correspond to the neurons of the input layer and the columns correspond to the 10 sub neural networks, each has 2 hidden units. The sparsity of the matrix demonstrates the sparse interaction between the neurons of the input layer.

hidden layers.

Implementation Details: Our Bayesian Neural Network is a sum of 10 small neural networks and each small network consists of 3 layers. An input feature vector is passed through 10 sub-neural networks followed by addition operation to give a final scalar output. For each sub-neural network, we use 2 to 5 neurons for the first hidden layer and 5 to 20 neurons for the second hidden layer. We train the Bayesian Neural Network with batch size $= 100$ and 0.01 initial learning rate with exponential decay until the validation error converges. In order to pick sparse interpretable variables, we impose group Lasso for the first layer with respect to each input neuron, which associates with sub neural network. The group Lasso penalty hyper-parameter depends on the sparsity and addition structure. In our experiments, it ranges from 0.001 to 1.0.

AddNN learns the sparse additive structure: To show our Bayesian additive neural network can learn the sparse addition structure of the function and the interaction, we provide one example of learning Friedman function f_1. We plot the learned matrix of the input layer and the first hidden layer, which can be seen in Fig 3.

The complete comparison for benchmark datasets: We train a Bayesian additive neural network on benchmark datasets and evaluate each model on all benchmark datasets. Table 4 shows that our proposed model (compact) offers favorable performance comparing with other methods for prediction accuracy. Due to space limitation, we only include the MC dropout baseline in the main body.

Table 4: Average test performance in RSME and Standard Errors for AddNN(ours), dropout uncertainty (MC dropout), deep Gaussian process (DGP 5) and probabilistic back-propagation(PBP) on benchmarks. Dataset size(N) and input dimensionality(Q) are also given.

	N	Q	AddNN	MC dropout[12]	DGP 5 [22]	PBP[16]
Boston	506	13	3.03 ± 0.12	2.97 ± 0.19	2.92 ± 0.17	3.01 ± 0.18
Concrete	1030	8	5.18 ± 0.14	5.23 ± 0.12	5.65 ± 0.10	5.67 ± 0.09
Energy	768	8	0.65 ± 0.03	1.66 ± 0.04	0.47 ± 0.01	1.80 ± 0.05
Kin8nm	8192	8	0.07 ± 0.00	0.10 ± 0.00	0.06 ± 0.00	0.10 ± 0.00
Naval	11934	16	0.01 ± 0.00	0.01 ± 0.00	0.00 ± 0.00	0.01 ± 0.00
Power	9568	4	4.04 ± 0.03	4.02 ± 0.04	3.68 ± 0.03	4.12 ± 0.03
Protein	45730	9	4.07 ± 0.01	4.36 ± 0.01	3.72 ± 0.04	4.73 ± 0.01
Wine	1599	11	0.66 ± 0.01	0.62 ± 0.01	0.63 ± 0.01	0.64 ± 0.01