Substituting Neural Networks for Gaussian Processes

Skyler D. Gray

A selected project submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Matthew J. Heaton, Chair
David B. Dahl
Jared D. Fisher

Department of Statistics

Brigham Young University

April 2022

ABSTRACT

Substituting Neural Networks for Gaussian Processes

Skyler D. Gray
Department of Statistics, BYU
Master of Science

The primary challenge of spatial statistics is to account for the inherent correlation that arises in the data due to proximity in sampling locations. To account for this correlation, spatial statisticians traditionally use the Gaussian process (GP). However, the GP is plagued by computational intractability, rendering it infeasible for use on large spatial data sets. On the other hand, neural networks (NNs) have arisen as a flexible approach for modeling nonlinear relationships. To date, however, neural networks have only been scarcely used for problems in spatial statistics. In this work, we demonstrate how to implement neural networks for spatial data. We compare the efficacy and predictive power of neural networks with that of GP approximations across a variety of big spatial data applications. Our results suggest that wide-layer NN representations of a GP are not the best predictive NNs for spatial data. However, our results indicate that fully-connected NNs approach the power of state-of-the-art, dataset-specific GP-approximated models. Next steps for this research include modeling binary response data, quantifying prediction uncertainty, and revising basis function expansion inputs.

Keywords: Big data, Gaussian process, fully-connected neural network, grid search

ACKNOWLEDGMENTS

This research work was the product of many hands. First and foremost, I want to thank God, my Heavenly Father, for His answers to my prayers. The inspiration and guidance from His hands made it possible to accomplish this project work. Second, I express my sincerest gratitude to my graduate advisor, Dr. Matthew Heaton, for his mentorship and contributions throughout this research process. His positivity, motivation, and vision are infectious and have been an inspiration to me in my education here at BYU. Next, I give special thanks to my wife, Emily Gray, for her support. Her endless love and encouragement have been invaluable throughout my Master's program. Lastly, I am filled with gratitude for the talented faculty in the Department of Statistics at BYU. The rigorous coursework and the selfless examples provided by this amazing group of scholars has truly enabled me to "go forth to serve."

CONTENTS

CHAPTER 1

INTRODUCTION

Spatial statistics concerns the analysis of spatially correlated data. Accounting for the correlation between observations can provide more accurate predictions of phenomena across diverse fields, including demographic trends, weather forecasts, ecological traits, and component time-to-failure analyses. Traditionally, spatial statisticians rely on GPs for modeling data across a continuous domain $\mathcal{D} \subset \mathbb{R}^d$ where d is typically 1 (time) or 2 (space) (see Heaton et al. 2019, for examples). In addition to accounting for geographical correlation, though, GPs are also implemented to account for various other distance-related correlations. Akhtar and Mian (2020) use GPs to recover spectral information from RGB images and account for the spatial correlation between image pixels. Twomey et al. (2019) use hierarchical GPs to detect outlying variable star light-curves. Zhang and Xu (2020) use a GP to more closely model the critical temperature of magnesium boride (MgB2) for different lattice parameters. Li et al. (2020) propose a one-dimensional GP, equivalent to time series analysis, to model lithium-ion battery state of health.

Recent applications have started to see complications in using GPs because of computational complexity and non-realistic correlation structures (e.g. assuming stationarity for simplicity when a process is, in fact, non-stationary). Traditionally fitting a GP to a large data set is prohibitively expensive, costing $\mathcal{O}(n^3)$ in computation time due to inverting an $n \times n$ matrix. However, researchers have developed multiple techniques to approximate GPs assuming certain model structures (Heaton et al. 2019; Liu et al. 2020). The most common technique to circumvent the computational complexity of GPs is to use low-rank basis functions (Sang and Huang 2012; Katzfuss 2017; Cressie and Johannesson 2008) or to insert sparsity into either the covariance matrix (Kaufman et al. 2008) or the precision matrix

(Datta et al. 2016a,b; Katzfuss and Guinness 2021). While the majority of these approaches focus on unrealistic stationary correlations, more recent work has been focused on utilizing these approaches for more realistic correlation structures (Huang et al. 2021b).

A recent connection discussed by Lee et al. (2017) has increased interest in using fully-connected neural networks (NNs) as one of these GP approximations. NNs are flexible model frameworks that "learn" patterns through an optimization algorithm that repeatedly iterates through the data. These models have countless realms of application, including predicting object distance, short-term rainfall forecasts, business financial distress, and chili plant disease classification (Mesa et al. 2019; Zhang et al. 2018; El Bannany et al. 2021; Nuanmeesri and Sriurai 2021). Because of their ability to capture complex relationships and ease computing via batching, NNs may be an interesting solution for modeling correlated in data.

The chief objective of this project is to understand the ability and limitations of using fully connected NNs to approximate GPs when applied to an array of different spatial data problems. If NNs can be used to reasonably model the correlation in spatial data, modeling complex data structures will be much more feasible for those not specializing in spatial statistics. By measuring performance metrics of both NNs and GPs (whether approximated via another method or exact), this project investigates the extent to which a NN can imitate a GP.

The remainder of this project is outlined as follows. Chapter 2 provides a review of key ideas of GPs and NNs as well as discusses the potential link between the two. Chapter 3 implements a NN approach to modeling various spatial datasets and compares its performance to using a GP. Chapter 4 draws conclusions and outlines areas for future work.

CHAPTER 2

METHODS

2.1 Gaussian Processes

Let $Y(\mathbf{s}) \in \mathbb{R}$ be a response variable measured at spatial location $\mathbf{s} \in D \subset \mathbb{R}^d$ where, for purposes of this project, we focus on $d = 2$. The surface $Y(\mathbf{s})$ follows a GP if, for any finite set of locations $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$, the vector $\mathbf{Y} = (Y(\mathbf{s}_1), \ldots, Y(\mathbf{s}_n))' \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where $\mathcal{N}(\boldsymbol{m}, \boldsymbol{S})$ denotes a multivariate Normal distribution with mean vector $\boldsymbol{m}$ and covariance matrix $\boldsymbol{S}$. The mean vector $\boldsymbol{\mu} = (\mu(\boldsymbol{s}_1), \ldots, \mu(\boldsymbol{s}_n))'$ denotes the mean at each of the n locations and is typically taken to be a linear combination of covariates such that $\mu(\boldsymbol{s}) = \boldsymbol{x}'(\boldsymbol{s})\boldsymbol{\beta}$ where $\boldsymbol{x}(\boldsymbol{s}) = (1, x_1(\boldsymbol{s}), \ldots, x_Q(\boldsymbol{s}))'$ is a vector of Q covariates plus a constant term (intercept). The covariance matrix

$$\boldsymbol{\Sigma} = \{K(\boldsymbol{s}_i, \boldsymbol{s}_j \mid \boldsymbol{\phi})\}_{i,j=1}^{n} \tag{2.1}$$

where $K(\boldsymbol{s}, \boldsymbol{s}' \mid \boldsymbol{\phi})$ is a covariance (or, in machine learning terminology, a kernel) function that induces correlation between location $\boldsymbol{s}$ and $\boldsymbol{s}'$ and includes unknown parameters $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_J)'$. Most commonly, $K(\boldsymbol{s}, \boldsymbol{s}')$ is of the Matérn class of stationary covariance functions such that

$$K(\boldsymbol{s}_1, \boldsymbol{s}_2 \mid \boldsymbol{\phi}) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{\|\mathbf{s}_1 - \boldsymbol{s}_2\|}{\rho} \right)^{\nu} K_\nu \left(\sqrt{2\nu} \frac{\|\mathbf{s}_1 - \boldsymbol{s}_2\|}{\rho} \right) \tag{2.2}$$

where $\boldsymbol{\phi} = (\sigma^2, \nu, \rho)$, σ^2 is the spatial variance parameter, ν is the process smoothness, and ρ is referred to as a spatial range parameter. Lastly, K_ν is the modified Bessel function of the second kind. While the Matérn covariance function is the most commonly used, for the purposes of exposition, we will assume that $K(\cdot, \cdot \mid \boldsymbol{\phi})$ is any general, positive definite function.

GPs are used in spatial statistics for both inferential purposes as well as prediction (Gelfand and Schliep 2016). On the inferential side, estimates of unknown parameters are typically obtained via maximum likelihood where the likelihood is given by

$$\mathcal{L} \propto |\boldsymbol{\Sigma}|^{-1/2} \exp\left\{-\frac{1}{2}(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})\right\}. \tag{2.3}$$

The advantage of using maximum likelihood is that the correlation structure of the data is built in to the associated maximum likelihood estimates. However, the likelihood in (2.3) reveals the associated challenge of using GPs. Specifically, the presence of $|\boldsymbol{\Sigma}|$ and $\boldsymbol{\Sigma}^{-1}$ in the likelihood function require $\mathcal{O}(n^3)$ operations which are prohibitively slow for large n.

Prediction via GPs occurs by exploiting the fact that conditional distributions of a multivariate normal distribution are normal. That is, let $\boldsymbol{Y}_p$ be a finite vector of yet-to-be observed values of the response value that we wish to obtain predictions for. Under the GP assumption, the joint distribution of $(\boldsymbol{Y}, \boldsymbol{Y}_p)$ is

$$\begin{pmatrix} \boldsymbol{Y} \\ \boldsymbol{Y}_p \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{X}\boldsymbol{\beta} \\ \boldsymbol{X}_p\boldsymbol{\beta} \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma} & \boldsymbol{\Sigma}_{op} \\ \boldsymbol{\Sigma}_{po} & \boldsymbol{\Sigma}_p \end{pmatrix}\right) \tag{2.4}$$

where $\boldsymbol{X}_p$ is the matrix of covariates for $\boldsymbol{Y}_p$, $\boldsymbol{\Sigma}_{op} = \boldsymbol{\Sigma}_{po}'$ is the covariance between $\boldsymbol{Y}$ and $\boldsymbol{Y}_p$, and $\boldsymbol{\Sigma}_p$ is the covariance matrix of $\boldsymbol{Y}_p$. Under this joint distribution, the conditional distribution of $\boldsymbol{Y}_p$ given $\boldsymbol{Y}$ is

$$\boldsymbol{Y}_p \mid \boldsymbol{Y} \sim \mathcal{N}\left(\boldsymbol{X}_p\boldsymbol{\beta} + \boldsymbol{\Sigma}_{po}\boldsymbol{\Sigma}^{-1}(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}), \boldsymbol{\Sigma}_p - \boldsymbol{\Sigma}_{po}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_{op}\right) \tag{2.5}$$

which not only yields a point prediction $(\boldsymbol{X}_p\boldsymbol{\beta} + \boldsymbol{\Sigma}_{po}\boldsymbol{\Sigma}^{-1}(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}))$ but also gives an associated measure of uncertainty associated with this prediction via the diagonal elements of $\boldsymbol{\Sigma}_p - \boldsymbol{\Sigma}_{po}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_{op}$.

2.2 Neural Networks (Multi-layer Perceptrons)

The NNs considered in this research are formally known as fully-connected NNs or multilayer perceptrons. These NNs are composed of three types of layers: the input layer, hidden layers,

and the output layer. Let $\boldsymbol{x}_\ell(\boldsymbol{s})$ be the neuron values at layer ℓ which has length P_ℓ (i.e. is of dimension $P_\ell \times 1$). The input layer, which we define as layer 0, defines $\boldsymbol{x}_0(\boldsymbol{s}) = (\boldsymbol{x}(\boldsymbol{s}),\ \boldsymbol{g}(\boldsymbol{s}))'$ as the the set of covariates $\boldsymbol{x}(\boldsymbol{s})$ excluding the constant term and joined with $\boldsymbol{g}(\boldsymbol{s})$, a transformation of the spatial information $\boldsymbol{s}$. These transformations of the location information can be an identity function transformation or other transformation functions, such as just s_1^2, to increase NN flexibility.

Under a multilayer perceptron, $Y(\boldsymbol{s}) = f_{L+1}(b_{L+1} + \boldsymbol{W}_{L+1}\boldsymbol{x}_L(\boldsymbol{s})) + \epsilon(\boldsymbol{s})$ where $\epsilon(\boldsymbol{s})$ is an error term,

$$\boldsymbol{x}_\ell(\boldsymbol{s}) = f_\ell(\boldsymbol{W}_\ell \boldsymbol{x}_{\ell-1}(\boldsymbol{s}) + \boldsymbol{b}_\ell), \tag{2.6}$$

L is the number of hidden layers in the model, $\boldsymbol{W}_\ell$ is a P_ℓ x $P_{\ell-1}$ matrix of weights, $\boldsymbol{b}_\ell$ is a $P_\ell \times 1$ matrix of bias weights (intercepts), and $f_\ell(\cdot)$ is an element-wise non-linear transformation, referred to as an activation function, that accounts for non-linear relationships between the inputs and the outputs (i.e. $\boldsymbol{x}(\boldsymbol{s})$ and $Y(\boldsymbol{s})$). While there are many possible choices of activation functions (see Ramachandran et al. 2017; Nwankpa et al. 2018, for a discussion), the transformation function used throughout this paper for $\ell = 1, \ldots, L-1$ is

$$f_\ell(x) = I(x > 0)x, \tag{2.7}$$

where $I(\mathcal{A})$ is an indicator for the set $\mathcal{A}$ (this activation function is referred to as a Rectified Linear Unit (ReLU)) and $f_{L+1}(x) = x$ is the identity activation function so as to match the support of $Y(\vec{s})$ at the output layer. We hold the activation functions constant for our analysis because, generally, each of the popular activation functions reach a similar predictive power (Ramachandran et al. 2017; Nwankpa et al. 2018), though for our purposes with fully-connected NNs, ReLU tends to achieve that predictive accuracy faster.

Because of the high dimensionality of the parameter space, NNs are trained via a form of stochastic gradient descent (SGD). This is realized through training the NNs on batches (i.e. subsets) of data at a time to minimize some loss function. At the beginning of training, randomly initialize the NN weight parameters. Then, feed a batch of training

data through the NN. Next, calculate the loss for each observation using the predicted and true response values. Compute the gradient of the loss function for each observation. This is called backpropagation and is very easy to compute using the chain rule. Then calculate the average gradient across the batch of observations. Lastly, update the weight parameters using the following:

$$\omega_{j+1} = \omega_j - \delta \nabla \mathcal{L}(\omega_j), \tag{2.8}$$

where ω_j represents the current weights, δ is the learning rate or "step size," and $\nabla \mathcal{L}$ is the gradient of the loss function. Continue iteratively training the NN until all observations have been included in a batch.

Given the simple framework for NNs, its advantages and subsequent popularity is apparent. First, modeling a multivariate continuous, binary or categorical response variable (or a mixture of them) is as simple as adjusting the number of output neurons and the associated loss function to match the corresponding response variable(s). Comparatively, GPs by definition are only for continuous response variables. Although they have been used in multivariate settings via a generalized linear model framework (Diggle et al. 1998, 2003), this approach adds additional complexity in model fitting for GPs. Second, NNs are also more computationally efficient. As described above, the computations involved are quick to compute and completely avoid the need to deal with large matrices. Thus, NNs have better scaling properties for many applications.

While NNs have many advantages, they are notorious for overfitting. As such, users of these models must be aware of and able to mitigate for model overfitting. There are several ways to prevent overfitting, such as decreasing the learning rate, using dropouts, simplifying the model, early stopping, using regularization in the loss function, and data augmentation. Another disadvantage of using NNs is that parameters are not interpretable due to the series of non-linear transformations via the activation functions. However, techniques such as partial dependence plots and feature importance measures have been developed to provide some interpretation of covariate effects (Molnar et al. 2021).

2.3 Using NNs in place of GPs

Work by Neal (1994) and Lee et al. (2017) has derived equivalence between infinitely wide NNs and GPs. While Neal (1994) and Lee et al. (2017) appeal to the central limit theorem to show equivalence, given their recently popularity in the literature (see Cressie and Johannesson 2008; Katzfuss 2017), we will argue for their equivalence here using a basis function representation of a GP. For the basis function argument, recall that the Karhunen-Loéve theorem states that a GP $Y(\boldsymbol{s})$ can be represented as a linear combination of basis functions as

$$Y(\boldsymbol{s}) = \lim_{P\to\infty} \sum_{p=1}^{P} e_p(\boldsymbol{s})\theta_p \tag{2.9}$$

where $e_p(\boldsymbol{s})$ are orthogonal eigenfunctions and θ_p are independent, zero mean Gaussian random variables with variances s_p (Cressie and Wikle 2015). The equivalence of the GP and a NN is seen by noting that the NN, at layer L, gives a set of bases $\boldsymbol{b}_{Lp} = (x_{Lp}(\boldsymbol{s}_1), \ldots, x_{Lp}(\boldsymbol{s}_n))'$ for $p = 1, \ldots, P_L$ where $x_{Lp}(\boldsymbol{s}_i)$ is the p^{th} neuron at layer L for observation $i = 1, \ldots, n$. That is, the NN model is

$$Y(\boldsymbol{s}) = b_{L+1} + \sum_{p=1}^{P_L} x_{Lp}(\boldsymbol{s}) w_{(L+1)p} \tag{2.10}$$

where $w_{(L+1)p}$ are independent weights. Orthogonalizing $\boldsymbol{b}_{L1}, \ldots, \boldsymbol{b}_{LP_L}$ (via, e.g., Gram-Schmidt orthogonalization) to $\boldsymbol{e}_{L1}, \ldots, \boldsymbol{e}_{LP_L}$ and allowing $P_L \to \infty$ demonstrates that an infinitely wide NN is equivalent to a GP.

Neal (1994) and Lee et al. (2017) establish that it is possible to fit a NN of infinite widths via Bayesian training using GP priors. Such infinitely wide NNs, however, are computationally infeasible for large datasets (with, say, more than 100,000 observations). Rather than using infinitely wide NNs, this project will follow Lee et al. (2017) in using NNs with large hidden layer widths to approximate GPs. This is similar to other basis function expansion approximations of GPs such as kernel convolutions (Higdon 1998), fixed

ranked kriging (Cressie and Johannesson 2008), predictive processes (Banerjee et al. 2008) or multi-resolution bases (Katzfuss 2017).

While using NNs to approximate GPs is similar to other basis function approaches, there are several distinct potential advantages to using NNs over the other approaches. First, the bases $\boldsymbol{b}_{L1}, \ldots, \boldsymbol{b}_{LP_L}$ used in NNs are estimated from the data rather than fixed *a priori*. Specifically, the $\boldsymbol{x}_\ell(\boldsymbol{s})$ in Equation (2.6) include unknown weights that are updated in the fitting process. This has the potential to create a more efficient set of basis functions for the process than is used in alternative basis function approaches. Further, the basis functions used in NNs are not constrained by stationarity or other simplifying assumptions. That is, the NN has the potential to adapt to any correlation structure present in the data.

Among the downsides to using a NN model are the sheer number of model hyperparameters to set. In addition to deciding crucial parameters of the fully-connected NN framework–such as the number of hidden layers and width of the network–it may be equally if not more important to set the weight initialization parameters, learning rate, or regularization parameters appropriately. This problem can be mitigated by performing a gridsearch across the parameter space for the NN setup. In addition to hyperparameter choice being difficult, fitting a NN to complex datasets may require several thousand neurons at each layer to learn the data structure. This would add to required computation time to fit a NN but, given the above strategies, these challenges are more surmountable than the computational challenges faced by full GPs.

CHAPTER 3

APPLICATIONS

3.1 NN Fitting Approach

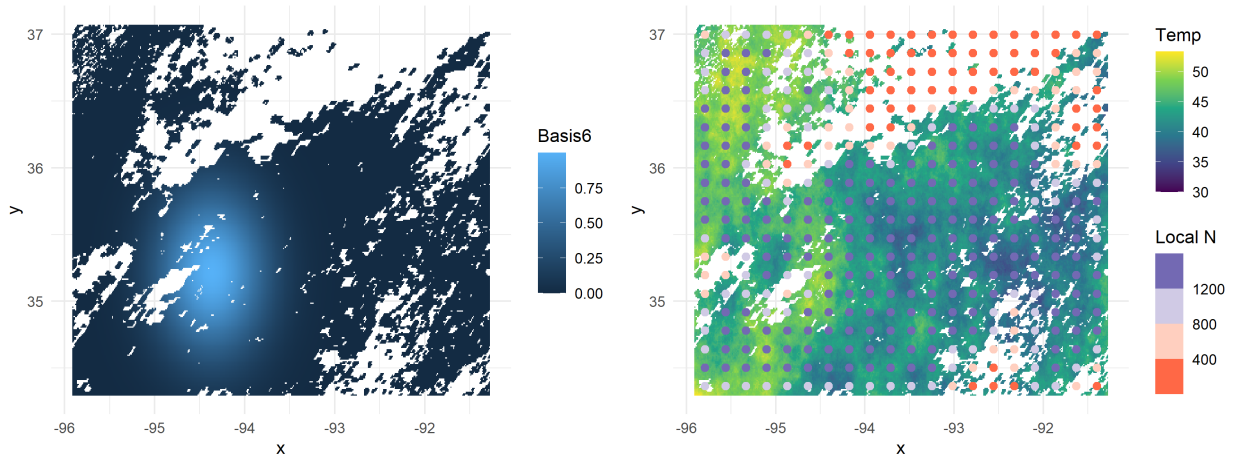
The NNs fit to the data in this research are strictly fully-connected NNs. Generally, we configure three types of parameters to find the optimal NN model setup for predictions across multiple datasets. These configuration types are NN input layer, structure design (how wide and deep the network is, how network weights are initialized, etc.), and optimization parameters (choice of optimizer algorithm, learning rate, learning rate decay, weight decay, dropout rate, loss function, etc.).

The datasets involved in this research have only x-y or latitude-longitude inputs. Using these coordinates, we explore NN performance for four different input layer configurations, one being the untransformed x-y values (Raw) and three others being basis function expansions. Of the basis function expansions, the first is a simple quadratic transformation (Trans) for both location values to create a 4×1 input layer of (x, x^2, y, y^2). The last two are radial basis function expansions of the location variables. Let $\boldsymbol{a}_i$ be the location of knot $i = 1, ..., A$, where A is the number of knots in the radial basis function expansion. Additionally, let

$$\theta = 2 \min_{i \neq j} ||\boldsymbol{a}_i - \boldsymbol{a}_j|| \tag{3.1}$$

represent twice the minimum distance between basis knots and

$$D = \begin{pmatrix} \frac{||\boldsymbol{s}_1 - \boldsymbol{a}_1||}{\theta} & \frac{||\boldsymbol{s}_1 - \boldsymbol{a}_2||}{\theta} & \cdots & \frac{||\boldsymbol{s}_1 - \boldsymbol{a}_A||}{\theta} \\ \frac{||\boldsymbol{s}_2 - \boldsymbol{a}_1||}{\theta} & \frac{||\boldsymbol{s}_2 - \boldsymbol{a}_2||}{\theta} & \cdots & \frac{||\boldsymbol{s}_2 - \boldsymbol{a}_A||}{\theta} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{||\boldsymbol{s}_n - \boldsymbol{a}_1||}{\theta} & \frac{||\boldsymbol{s}_n - \boldsymbol{a}_2||}{\theta} & \cdots & \frac{||\boldsymbol{s}_n - \boldsymbol{a}_A||}{\theta} \end{pmatrix} \tag{3.2}$$



(a) (b)

Figure 3.1: (a) An example of a radial basis function expansion on a knot located at about (-93, 35.25). The values closest to the knot have higher values and past some distance threshold, all observations have a value of 0. (b) 400 radial basis knots evenly spaced across the training data. Not all basis transformations have the same amount of local observations to train the data across the knots, so some knots should be dropped from the basis function transformation before fitting.

be an $n \times A$ adjusted distance matrix from every observation location to every basis knot. The radial basis function we used is a revision of Wendland's function for 2D data,

$$\phi(d) = \mathbf{1}(d \leq 1)\frac{(1-d)^6(35d^2+18d+3)}{3}, \tag{3.3}$$

where d is a distance metric between an observation and a basis knot. The radial basis function expansions are, then, an element-wise transformation of D into $\phi(D)$.

We explore the performance of a coarse 16-knot basis expansion (CB) as well as a multi-resolution basis expansion (MRB) that includes both the coarse 16-knot expansion as well as a 400-knot expansion. The optimal NN configuration of these four different input structures was calculated using both a custom-designed grid search as well as one imitating that of Lee et al. (2017) to mimic a GP. Figure 3.1a shows an example of what a transformation of the location data looks like for radial basis function expansions of the 150K-observation simulated temperature dataset. Sixteen knots were evenly spaced across the location space. The 16-variable transformation represents distances from local observations

to 16 knots evenly spaced across the rectangular region of the sample space, where higher values are associated with observations being closer to a given knot.

It is worth noting that radial basis function expansions are only useful if their local sample size is large enough. Figure 3.1b shows the location of all four hundred knots in a radial basis function expansion to 400 areas of local correlation. Given the granularity of this local correlation structure, a handful of knots have much fewer observations relative to the other knots. Given a small enough local sample size, the basis-transformed variables of some new predictors may all be 0. Fitting a NN to this data would likely lead to poor fitting and thus extreme predictions in the areas with little data. To prevent this, we only keep the basis transformations of those knots for which the local sample size is greater than 30 and whose maximum predictor value is 0.75 or greater. The local sample size is calculated as the sum of all n transformations of the given basis $\boldsymbol{a}_i$.

Grid Search

The remaining two NN configuration types – its structure design and optimization parameters – are chosen via grid search to increase objectivity and the dynamic capabilities of model fitting across each dataset. The first of the two grid searches we implemented is our custom grid search. This grid explores every combination of different parameter values with 800K or less weight parameters between all hidden layers. The hyperparameters included in the grid search included the number of hidden layers $\{2^0,\ 2^1,\ \ldots,\ 2^4\}$, layer width $\{2^3,\ 2^6,\ 2^7,\ \ldots,\ 2^{11}\}$, batch size $\{2^4,\ 2^5,\ \ldots,\ 2^8\}$, learning rate decay $\{0,\ \frac{0.01}{\lfloor N_{train}/batch\_size \rfloor}\}$, and dropout rate $\{0,\ 0.1\}$. In total, 480 NNs are fit per input type for the custom grid search method. For each NN fit, the learning rate was set to a constant 0.001.

In addition to using a custom grid search, we imitate the grid search used in Lee et al. (2017) (which we will refer to as the Lee2018 grid search). The NNs fit using this grid search for each dataset are optimized through a random search of 50 trials on several initialization and learning parameters for each choice of (depth, width) for our NN frame-

work. The following NN hyperparameters are randomly sampled for each trial: learning rate, weight decay, the weight parameter sampling distribution standard deviation σ_w, the bias parameter sampling distribution standard deviation σ_b, and the batch size. The continuous hyperparameters are sampled from a uniform distribution with varying ranges. Learning rates are sampled within (10^{-4}, 0.2) on the log-scale, weight decay within (10^{-8}, 1) on the log-scale, σ_w within (0.01, 2.25) and σ_b within (0, 1.5). Batch size was sampled with even probability from among 16, 32, 64, 128, 256. In total, 950 NNs are fit per input type using the Lee2018 grid search.

NN Constants and Loss Function

The NNs across all input types and grid searches use the ReLU activation function for every layer except the output layer. Additionally, the loss function used for fitting NNs was the mean squared error (MSE) but is reported as root mean squared error (RMSE) for interpretability. Where hyperparameters are not explicitly mentioned, such as initialization parameters for the NNs fit in the custom grid search, the default parameter settings of the Keras library in R are used.

What follows is an introduction to each example dataset, a quantitative comparison between the eight best-fit NNs, and an evaluation of the best fit NNs across datasets. Table 3.1 shows the validation performance of the best NN in each grid-input model combination. Table 3.2 contains test metrics on the Raw input type and the best of the basis function expansion input types (Basis) for both the custom and Lee2018 grid searches as well as the state-of-the-art (SoTA) model.

It should be noted that some of the NN parameterizations produced by the Lee2018 grid search did not fit to the data properly and were excluded from the analysis. If the randomly sampled learning rate was in the range of (e^{-4}, e^{-1}), the model potentially failed fitting, likely due to the unstable, large changes in weights with each post-batch update.

		Custom Grid				Lee2018 Grid			
Dataset	Metric	Raw	Trans	CB	MRB	Raw	Trans	CB	MRB
150K Quant	RMSE	**1.27**	1.26	**1.24**	1.30	**1.28**	1.27	**1.20**	1.25
1Mil-Gaus Quant	RMSE	**0.026**	0.028	0.018	**0.0078**	**0.053**	0.042	0.032	**0.014**
1Mil-NG Quant	RMSE	**0.15**	0.13	0.11	**0.08**	**0.18**	0.18	0.14	**0.09**

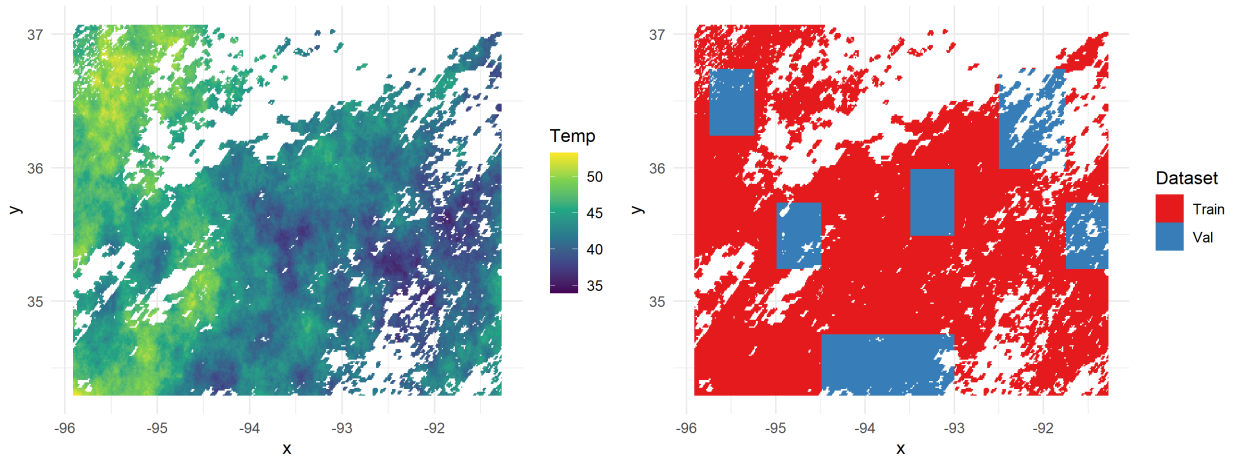
Table 3.1: Validation set performance across best untransformed input NNs (Raw) and basis function expansion input type NNs (Trans, CB, and MRB). The untransformed input type is trained for test performance along with the best performing basis input type (bolded).

3.2 Example 1: 150K Simulated Temperatures

This dataset was generated using real daytime land surface temperature readings from the MODIS satellite on August 4, 2016 and was included as a comparative dataset in Heaton et al. (2019) allowing us to compare the performance of the NN to the state-of-the-art alternatives in spatial kriging. This dataset was simulated by fitting a GP to a random sample of 2,500 observations, simulating 150K from the fitted GP and split into roughly a 100K-50K train-test split. The withheld test data mimics cloud cover that inhibits measurement of land surface temperature (see Figure 3.2a). Neither the train nor test set are distributed normally across the x-y coordinate space because the cloud cover blocks out groups of measurements in the location space. This dataset provides us the opportunity to evaluate NN performance relative to GP approximations in long range predictions.

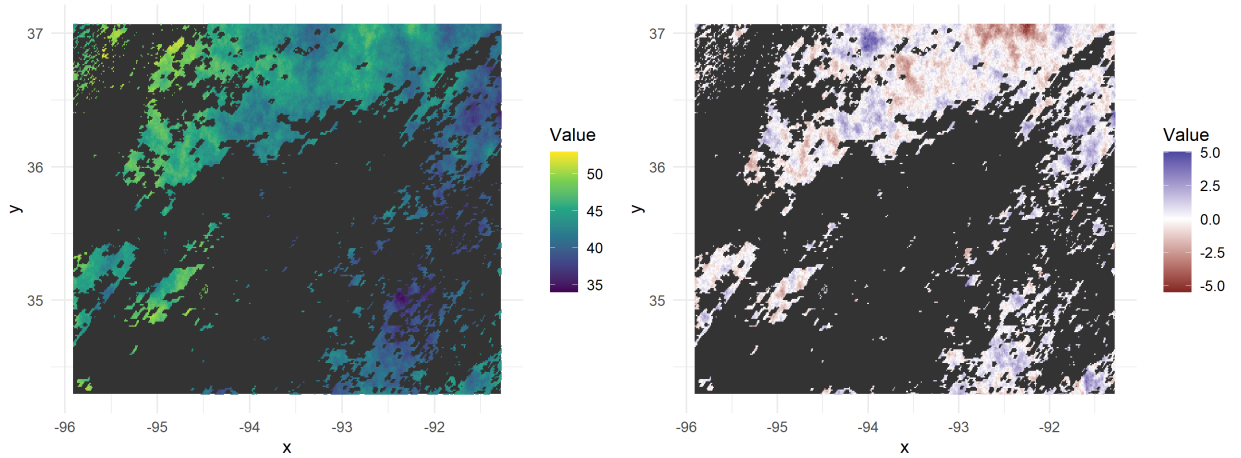
In an effort to train the data to predict in whole regions with no information, we manually divided about 20% of the data from the training set into a validation set (see Figure 3.2b). The best-performing validation RMSE among the input types of both grid searches was the CB input type with a custom grid RMSE of 1.24 and a Lee2018 grid RMSE of 1.20.

Among the four final models we fit, the Lee2018 CB NN performed the best with a test RMSE of 1.11, meaning that our temperature estimations for missing data were off by about 1.1 degrees on average. The prediction error for this model is shown in Figure 3.3b. Given the temperature ranged from about 35 to 55 degrees, one degree off on average is a



(a) (b)

Figure 3.2: (a) The 150K training data. The raster image is a 100×100 pixel representation of the data, so the value of all point coordinates that fall within the same pixel area are averaged together. (b) Whole regions of the training data were withheld for use as validation data.



(a) (b)

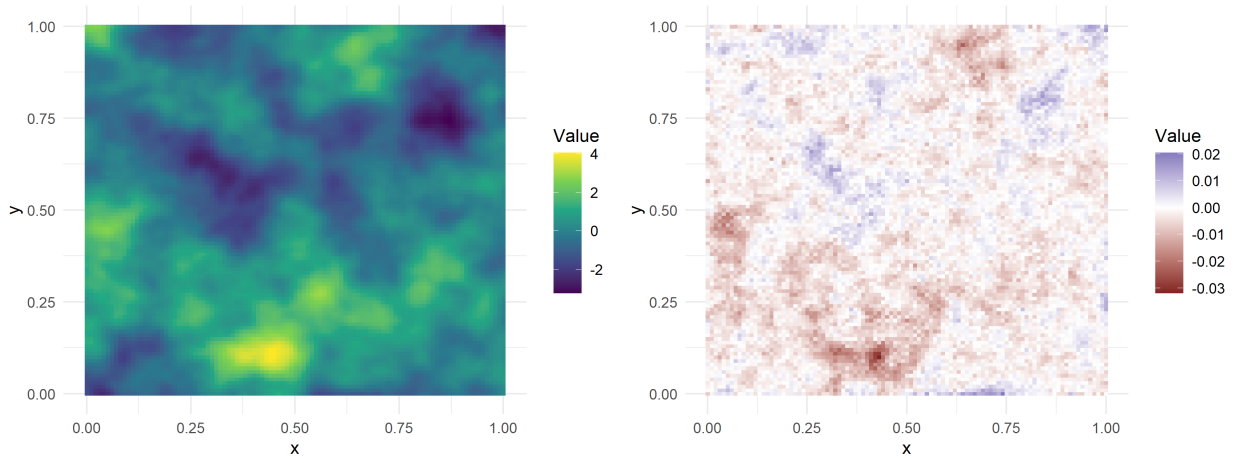
Figure 3.3: (a) The 50K test data. The area where only training data exist is colored gray to assist in visualizing test predictions. (b) Test prediction error for the 150K simulated temperature data.

useful level of precision. Our model did poorly predicting in particularly cool or warm spots far from local data points, which is what we would expect from extrapolation.

3.3 Example 2: 1 Million Gaussian Quantitative

This 1-million observation dataset comes from the Huang et al. (2021a) competition comparing the predictive performance of many spatial methods. The dataset was generated by a GP and was split into training and test sets of size 900K and 100K, respectively (see Figure 3.4a). Unlike our first example, both the training and test data are uniformly distributed across the x-y space, where $x, y \in (0, 1)$. Evaluating the predictive performance of our model on this dataset illustrates how well we can interpolate using a NN approach in addition to demonstrating the feasibility of fitting the model to all the available data.

Under both the custom and Lee2018 grid searches, the MRB input type performed the best. The custom grid search outperformed Lee2018 with an RMSE of 0.0078 as opposed to 0.014 (see Table 3.2). The performance of the MRB fit to the custom grid search also dominated among the four final models tested with a test RMSE of 0.0066. Figure 3.4b shows that the area of worst prediction performance was in the peak on the lower center area of the sample space. While this is our best performing model, the error across the prediction space has several spaces consistently predicted low or high, suggesting that the model has not fully captured the correlation between the observations.



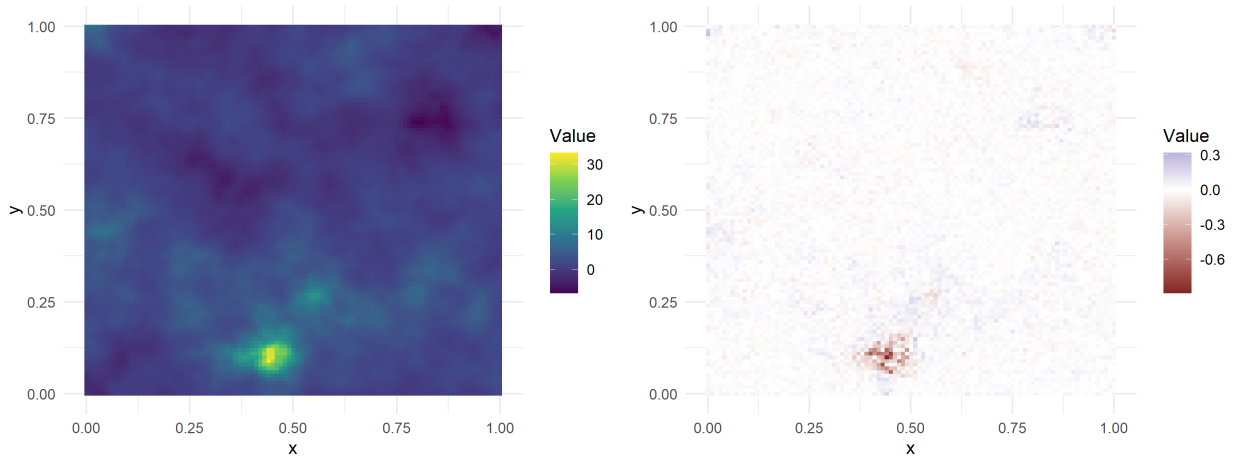
(a) (b)

Figure 3.4: (a) The 900K Gaussian training data. (b) The average prediction error of the best performing model, the MRB fit with the custom grid search.

3.4 Example 3: 1 Million Non-Gaussian Quantitative

The data from this example comes from the same source as Example 2 (Huang et al. 2021a). The non-Gaussian dataset was generated by Tukey g-and-h random fields and was split into training and test sets of size 900K and 100K, respectively (see Figure 3.5a). The training and test data are uniformly distributed across the x-y space, where $x, y \in (0, 1)$. Unique to this dataset, however, is its non-Gaussian nature, something we should catch with our flexible NN approach.

Under both the custom and Lee2018 grid searches, the MRB input type performed the best. Both grid searches performed similarly well with the custom search achieving a validation RMSE of 0.08 and Lee2018 of 0.09 (see Table 3.2). The next best performers in terms of validation RMSE were the coarse basis, transformed inputs, and raw inputs with their best RMSEs of 0.11, 0.13, and 0.15, respectively. The MRB fit using the custom grid search also performed best with a test RMSE of 0.072. Figure 3.5b shows our model had the hardest time approximating the sharp peak in the data. Overall, it underestimated how high the peak value was. This may have been caused by early stopping, which may have not given the NN enough time to learn the nuances of the data.



(a) (b)

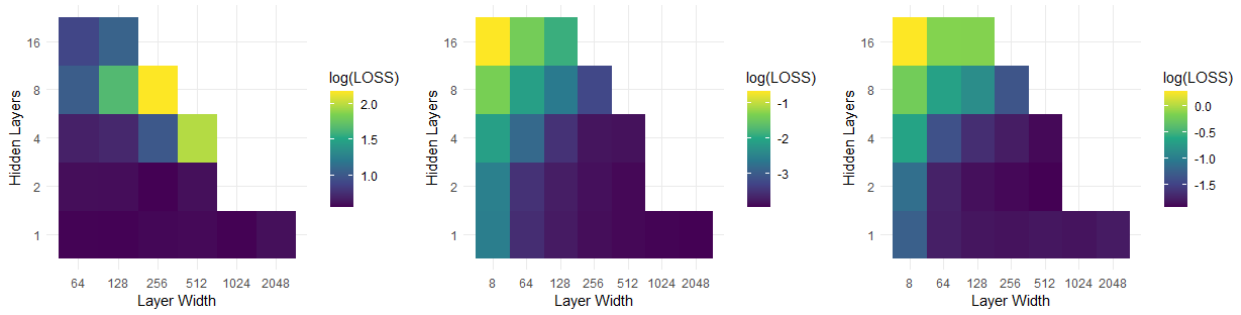
Figure 3.5: (a) The 900K non-Gaussian training data. (b) The average prediction error of the best performing model, the MRB fit with the custom grid search.

3.5 Discussion

The test RMSEs for our best performing models are similar to their validation RMSE analogs, indicating that our validation sets were useful for tuning NN model hyperparameters.

Our results indicate that the optimal NN (depth, width) structure varied among both datasets and input types. Within the top-performing input-grid optimization, however, we observe that shallow NNs tend to perform better if they are sufficiently wide. Figures 3.6a, 3.6b, and 3.6c represent the grid search results for the grid and input type with best performing validation RMSE. Given the hidden layers are at least 64 neurons wide, we see that one or two hidden layers tends to perform similarly well in predicting new observations of data. It is interesting to note that while a fully-connected NN equals a GP only as the width of its hidden layers approaches infinity, we see a similar predictive performance across NNs with a width of at least 64.

Our best models did not perform as well as the state-of-the-art models fit to these datasets. Our best performing model relative to SoTA is the simulated temperature model at about 1.5 times SoTA RMSE. The 1-million non-Gaussian dataset comes second at about



(a) (b) (c)

Figure 3.6: The best validation loss of each NN in a grid search for a given input type is aggregated in these plots. Because of right skewness, the losses were first log-transformed and then averaged across each hidden layer depth-width combination. The plots represent (a) 150K CB input, Lee2018 grid; (b) 1Mil Gaussian MRB input, custom grid; and (c) 1Mil non-Gaussian MRB input, custom grid.

3 times SoTA RMSE and the 1-million Gaussian dataset performed the worst with about 7 times SoTA RMSE.

Between the two grid searches, our custom grid search performed better on average than the Lee2018 grid search. Among the three datasets, the Lee2018 grid search outperformed our custom grid search once with a test RMSE 4% lower than the custom grid search's best model. However, the custom grid search outperformed the Lee2018 grid search on the other two datasets, the largest difference in test RMSE being 63% lower compared to the optimized Lee2018 model. The Lee2018 grid explored different hyperparameters than our custom grid search, including randomly sampling the learning rate, weight decay, the variance of the weights, and the variance of the hidden layer bias terms. While this grid search may have considered a larger hyperparameter space, the hyperparameter tuning it considered did not appear to give it any practical advantage in predictions over our custom grid search optimization.

		Custom Grid		Lee2018 Grid		SoTA
Dataset	Test Metric	Raw	Basis	Raw	Basis	
150K Quant	RMSE	1.20	1.16	1.94	**1.11**	0.83
1Mil-Gaus Quant	RMSE	0.029	**0.0066**	0.063	0.018	0.00095
1Mil-NG Quant	RMSE	0.120	**0.072**	0.181	0.087	0.021

Table 3.2: Performance of custom and Lee2018 grid search-optimized NN setups with state-of-the-art (SoTA) for reference. Only the untransformed input structure (Raw) and the basis function expansion input structure with the best validation performance (be it the quadratic, coarse radial basis, or multi-resolution basis function expansion) are evaluated on the test data.

CHAPTER 4

CONCLUSIONS

The purpose of our research was to understand the advantages and limitations of using a fully-connected NN in place of a GP for big data scenarios. While, in theory, a NN with infinitely wide hidden layers is equivalent to a GP, our results suggest that wide-layer NN representations of a GP are not the best predictive NNs for spatial data. For both the Gaussian and non-Gaussian datasets, we did not achieve as accurate a predictive score as the state-of-the-art, dataset-specific GP-approximated models. However, our results are encouraging, as we approached a similar predictive power in one of the three datasets.

One caveat to our approach of comparing one model design with many others across several datasets is our approach is not specially fitted to any one dataset. Models tuned to data-specific phenomenon observed in data EDA have the ability to learn patterns found in the data. It would be interesting to further investigate NN modeling specifically for each dataset.

In addition to specializing model structures for each given dataset, there are several other potential avenues of further research. First, we wish to look at our NN model's performance for many other datasets. Second, our models do not currently incorporate any uncertainty in their predictions. Using the dropout rate to make multiple predictions of the same observation is one way to begin investigating the certainty of our model's predictions. Third, all investigated NN structures were limited to those with less than 800K parameters between the hidden layers. Investigating overparameterized models for datasets with more than a 800K observations would be interesting. Fourth, the simple quadratic basis function expansion could be extended to calculate other transformations, such as computing the sine or cosine. Fifth, closer evaluation of prediction error patterns within validation sets may

also provide insight on where to change model or grid search structures. Lastly, we used the early-stop technique to speed up grid search computations, but that may have inhibited the performance of our grid searches in finding the best model parameterizations. Increasing the requirements for early stopping may yield significantly improved predictive performance in NN models.

BIBLIOGRAPHY

Akhtar, N., and Mian, A. (2020), "Hyperspectral Recovery from RGB Images using Gaussian Processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42, 100–113.

Banerjee, S., Gelfand, A. E., Finley, A. O., and Sang, H. (2008), "Gaussian predictive process models for large spatial data sets," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70, 825–848.

Cressie, N., and Johannesson, G. (2008), "Fixed Rank Kriging for very large spatial data sets," *Journal of the Royal Statistical Society B*, 70, 209–226.

Cressie, N., and Wikle, C. K. (2015), *Statistics for spatio-temporal data*, John Wiley & Sons.

Datta, A., Banerjee, S., Finley, A. O., and Gelfand, A. E. (2016a), "Hierarchical nearest-neighbor Gaussian process models for large geostatistical datasets," *Journal of the American Statistical Association*, 111, 800–812.

—— (2016b), "On nearest-neighbor Gaussian process models for massive spatial data," *Wiley Interdisciplinary Reviews: Computational Statistics*, 8, 162–171.

Diggle, P. J., Ribeiro, P. J., and Christensen, O. F. (2003), "An introduction to model-based geostatistics," in *Spatial Statistics and Computational Methods*, Springer, pp. 43–86.

Diggle, P. J., Tawn, J. A., and Moyeed, R. A. (1998), "Model-based geostatistics," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 47, 299–350.

El Bannany, M., Khedr, A. M., Sreedharan, M., and Kanakkayil, S. (2021), "Financial Distress Prediction based on Multi-Layer Perceptron with Parameter Optimization." *IAENG International Journal of Computer Science*, 48.

Gelfand, A. E., and Schliep, E. M. (2016), "Spatial statistics and Gaussian processes: A beautiful marriage," *Spatial Statistics*, 18, 86–104, spatial Statistics Avignon: Emerging Patterns.

Heaton, M. J., Datta, A., Finley, A. O., Furrer, R., Guinness, J., Guhaniyogi, R., Gerber, F., Gramacy, R. B., Hammerling, D., Katzfuss, M., et al. (2019), "A case study competition among methods for analyzing large spatial data," *Journal of Agricultural, Biological and Environmental Statistics*, 24, 398–425.

Higdon, D. (1998), "A process-convolution approach to modelling temperatures in the North Atlantic Ocean," *Environmental and Ecological Statistics*, 5, 173–190.

Huang, H., Abdulah, S., Sun, Y., Ltaief, H., Keyes, D. E., and Genton, M. G. (2021a), "Competition on spatial statistics for large datasets," *Journal of Agricultural, Biological and Environmental Statistics*, 26, 580–595.

Huang, H., Blake, L. R., Katzfuss, M., and Hammerling, D. M. (2021b), "Nonstationary Spatial Modeling of Massive Global Satellite Data," *arXiv:2111.13428*.

Katzfuss, M. (2017), "A multi-resolution approximation for massive spatial datasets," *Journal of the American Statistical Association*, 112, 201–214.

Katzfuss, M., and Guinness, J. (2021), "A general framework for Vecchia approximations of Gaussian processes," *Statistical Science*, 36, 124–141.

Kaufman, C. G., Schervish, M. J., and Nychka, D. W. (2008), "Covariance tapering for likelihood-based estimation in large spatial data sets," *Journal of the American Statistical Association*, 103, 1545–1555.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. (2017), "Deep Neural Networks as Gaussian Processes," .

Li, X., Yuan, C., Li, X., and Wang, Z. (2020), "State of health estimation for Li-Ion battery using incremental capacity analysis and Gaussian process regression," *Energy*, 190.

Liu, H., Ong, Y.-S., Shen, X., and Cai, J. (2020), "When Gaussian Process Meets Big Data: A Review of Scalable GPs," *IEEE Transactions on Neural Networks and Learning Systems*, 31, 4405–4423.

Mesa, J., Vasquez, D. B., Aguirre, J. V., and Valencia, J. S. B. (2019), "Sensor fusion for distance estimation under disturbance with reflective optical sensors using multi layer perceptron (MLP)," *IEEE Latin America Transactions*, 17, 1418–1423.

Molnar, C., Freiesleben, T., König, G., Casalicchio, G., Wright, M. N., and Bischl, B. (2021), "Relating the Partial Dependence Plot and Permutation Feature Importance to the Data Generating Process," *arXiv preprint arXiv:2109.01433*.

Neal, R. M. (1994), "Priors for infinite networks (tech. rep. no. crg-tr-94-1)," *University of Toronto*.

Nuanmeesri, S., and Sriurai, W. (2021), "Multi-Layer Perceptron Neural Network Model Development for Chili Pepper Disease Diagnosis Using Filter and Wrapper Feature Selection Methods," *Engineering, Technology & Applied Science Research*, 11, 7714–7719.

Nwankpa, C., Ijomah, W., Gachagan, A., and Marshall, S. (2018), "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*.

Ramachandran, P., Zoph, B., and Le, Q. V. (2017), "Searching for activation functions," *arXiv preprint arXiv:1710.05941*.

Sang, H., and Huang, J. Z. (2012), "A full scale approximation of covariance functions for large spatial data sets," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74, 111–132.

Twomey, N., Chen, H., Diethe, T., and Flach, P. (2019), "An application of hierarchical Gaussian processes to the detection of anomalies in star light curves," *Neurocomputing*, 342, 152–163.

Zhang, P., Jia, Y., Gao, J., Song, W., and Leung, H. (2018), "Short-term rainfall forecasting using multi-layer perceptron," *IEEE Transactions on Big Data*, 6, 93–106.

Zhang, Y., and Xu, X. (2020), "Predicting doped MgB2 superconductor critical temperature from lattice parameters using Gaussian process regression," *Physica C-Superconductivity and Its Applications*, 573.