Journal of Data Science 20 (4), 493–511
October 2022
DOI: 10.6339/22-JDS1070
Data Science in Action

On the Use of Deep Neural Networks for Large-Scale Spatial Prediction

Skyler D. Gray[1], Matthew J. Heaton[1,*], Dan S. Bolintineanu[2], and Aaron Olson[3]

[1]*Department of Statistics, Brigham Young University, 2152 WVB, Provo, UT 84602, United States*
[2]*Fluid and Reactive Processes Department, Sandia National Laboratories, Albuquerque, NM 87185, USA*
[3]*Radiation Effects Theory Department, Sandia National Laboratories, Albuquerque, NM 87185, USA*

Abstract

For spatial kriging (prediction), the Gaussian process (GP) has been the go-to tool of spatial statisticians for decades. However, the GP is plagued by computational intractability, rendering it infeasible for use on large spatial data sets. Neural networks (NNs), on the other hand, have arisen as a flexible and computationally feasible approach for capturing nonlinear relationships. To date, however, NNs have only been scarcely used for problems in spatial statistics but their use is beginning to take root. In this work, we argue for equivalence between a NN and a GP and demonstrate how to implement NNs for kriging from large spatial data. We compare the computational efficacy and predictive power of NNs with that of GP approximations across a variety of big spatial Gaussian, non-Gaussian and binary data applications of up to size $n = 10^6$. Our results suggest that fully-connected NNs perform similarly to state-of-the-art, GP-approximated models for short-range predictions but can suffer for longer range predictions.

Keywords *big data; fully-connected neural network; grid search*

1 Introduction

Spatial statistics concerns the analysis of spatially correlated data. Accounting for the spatial 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, the go-to tool for spatial statisticians is the Gaussian process (GP). The inherent advantage of using a GP is the ability to leverage the spatial correlation in the data to generate a prediction at a new location (referred to as "kriging" in the spatial statistics literature). Such predictions are non-linear in space and, therefore, do not rely on traditional, perhaps limiting, assumptions.

Recent applications involving big data, however, have started to see complications in using GPs because of the 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 where n is the sample size. However, researchers have developed multiple techniques to approximate GPs and thereby still utilize their strengths for prediction purposes (see Heaton et al., 2019; Liu et al., 2020, for reviews). Such approximations include low-

*Corresponding author. Email: mheaton@stat.byu.edu.

Received July 28, 2022; Accepted September 27, 2022

rank basis functions (Sang and Huang, 2012; Katzfuss, 2017; Cressie and Johannesson, 2008a) or inserting sparsity into either the covariance matrix (Furrer et al., 2006; Kaufman et al., 2008) or the precision matrix (Datta et al., 2016a,b; Katzfuss and Guinness, 2021). While most of these approximations use unrealistic correlation structures, recent work has focused on utilizing these approaches for more realistic correlation structures (Huang et al., 2021b).

While the GP has been the favorite tool among statisticians, neural networks (NNs) have exploded in popularity in the computer science realm due to their ability to approximate nearly any function (Yarotsky, 2018). 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).

A recent connection discussed by Lee et al. (2018) and Matthews et al. (2018) states that the universal approximation theorem says a single hidden layer can approximate a GP as long as the number of hidden neurons goes to infinity. Deep neural nets, consequently, have the potential to move beyond GPs. Seminal work in this regard by Chen et al. (2022) outlines a general approach for the use of fully-connected NNs for spatial prediction. Lenzi et al. (2021) and Gerber and Nychka (2021) have used NNs to perform parameter estimation for GPs and then subsequently do plug-in prediction. Zammit-Mangion and Wikle (2020) use convolutional NNs to estimate parameters of a space-time process while Zammit-Mangion et al. (2021) use NNs as warping functions for non-stationary spatial models. Likewise, Sauer et al. (2022) use NNs in place of GP emulators for computer experiments with extensions by Sauer et al. (2022) which make the computation more scalable. We refer the reader to the most recent review paper by Wikle and Zammit-Mangion (2022) for a more thorough review.

Many of the above uses of NNs for traditional spatial problems are still plagued by computational issues or have not fully explored the use of NNs for large spatial prediction problems (e.g. Chen et al. 2022 only consider datasets as large at $n \approx 12,000$). To fill this research hole, the chief objective of this research is to further understand the ability and limitations of using fully-connected NNs to perform spatial prediction from large spatial datasets. Specifically, we wish to evaluate the use of NNs for Gaussian and non-Gaussian, as well as short and long range prediction problems from data on the order of $n \approx 10^6$. Our results suggest that such methods are computationally efficient but primarily accurate only when the data are dense (see Section 4 for more details).

The remainder of this paper is outlined as follows. Section 2 provides a review of key ideas of GPs and NNs as well as discusses the potential link between the two. Section 3 details our implementation of a NN and Section 4 compares the NN performance to using a GP. Section 5 draws conclusions and outlines areas for future work.

2 Methods

2.1 Gaussian Processes

Let $Y(\mathbf{s})$ be a response variable measured at spatial location $\mathbf{s} \in D \subset \mathbb{R}^d$ where, for purposes of this research, 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 $\mathbf{Y} = (Y(\mathbf{s}_1), \ldots, Y(\mathbf{s}_n))' \sim \mathcal{N}_n(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where $\mathcal{N}_n(\boldsymbol{m}, \boldsymbol{S})$ 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

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

where $K(\boldsymbol{s}_i, \boldsymbol{s}_j \mid \boldsymbol{\phi})$ 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, $K(\boldsymbol{s}_1, \boldsymbol{s}_2)$ is of the Matérn class of stationary covariance functions such that

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

where $\boldsymbol{\phi} = (\sigma^2, \nu, \rho)'$, K_ν is the modified Bessel function of the second kind, σ^2 is the spatial variance parameter, ν is the process smoothness, and ρ is referred to as a spatial range parameter. 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

$$\mathcal{L} \propto |\boldsymbol{\Sigma}|^{-1/2} \exp\left\{ -\frac{1}{2} (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})' \boldsymbol{\Sigma}^{-1} (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}) \right\}. \tag{3}$$

The advantage of using maximum likelihood is that the correlation structure of the data is built into the associated maximum likelihood estimates. However, the likelihood in (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 m-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

$$\begin{pmatrix} \boldsymbol{Y} \\ \boldsymbol{Y}_p \end{pmatrix} \sim \mathcal{N}_{n+m} \left(\begin{pmatrix} \boldsymbol{X}\boldsymbol{\beta} \\ \boldsymbol{X}_p\boldsymbol{\beta} \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma} & \boldsymbol{\Sigma}_{op} \\ \boldsymbol{\Sigma}_{po} & \boldsymbol{\Sigma}_p \end{pmatrix} \right) \tag{4}$$

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

$$\boldsymbol{Y}_p \mid \boldsymbol{Y} \sim \mathcal{N}_m \left(\boldsymbol{X}_p\boldsymbol{\beta} + \boldsymbol{\Sigma}_{po}\boldsymbol{\Sigma}^{-1}(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}), \boldsymbol{\Sigma}_p - \boldsymbol{\Sigma}_{po}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_{op} \right) \tag{5}$$

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 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 s_1^2, to increase NN flexibility.

Under a multilayer perceptron, $\mathbb{E}(Y(\boldsymbol{s})) = f_{L+1}(b_{L+1} + \boldsymbol{W}_{L+1}\boldsymbol{x}_L(\boldsymbol{s}))$ where

$$\boldsymbol{x}_\ell(\boldsymbol{s}) = f_\ell(\boldsymbol{W}_\ell \boldsymbol{x}_{\ell-1}(\boldsymbol{s}) + \boldsymbol{b}_\ell), \tag{6}$$

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. 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 $\ell = 1, \ldots, L$ is

$$f_\ell(x) = I(x > 0)x, \tag{7}$$

where $I(\mathcal{A})$ is an indicator for the set $\mathcal{A}$ (this activation function is referred to as a Rectified Linear Unit (ReLU)). The final activation function $f_{L+1}(x)$ is chosen so as to match the support of $Y(\mathbf{s})$ at the output layer. For example, if $Y(\mathbf{s})$ is real-valued, then f_{L+1} is typically the identity function while if $Y(\mathbf{s})$ is binary then f_{L+1} may be the sigmoid function. 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.

The unknowns (parameters) of the NN are the set of weights $\{\boldsymbol{W}_\ell\}$ and biases $\{\boldsymbol{b}_\ell\}$ for $l = 1, \ldots, L+1$. Because of the high dimensionality of the parameter space, NNs are trained via some form of gradient descent methodology (e.g. stochastic gradient descent or ADAM optimizers). We do not endeavor to cover gradient descent methods in detail here but, for purposes of this research, we mention two tuning parameters associated with these methods. First, the batch size is the size of the subsample of the data used at one training iteration. And, second, the learning rate is the step size used at each iteration.

Given the simple framework for NNs, its advantages and subsequent popularity is apparent for a few reasons. 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, the output activations and the associated loss function to match the corresponding response variable(s). Comparatively, GPs by definition are only for continuous response variables, though they have been used in multivariate settings (Genton and Kleiber, 2015) via a generalized linear model framework (Diggle et al., 1998, 2003). When used outside the continuous response variable realm, GPs have additional complexity in model fitting. Second, NNs are far more computationally efficient than GPs. As described above, the computations involved completely avoid the need to deal with large matrices by utilizing batch subsampling.

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 (Jabbar and Khan, 2015). 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. A second 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 as Approximations of GPs

Work by Neal (1994) and Lee et al. (2018) has derived equivalence between infinitely wide NNs and GPs. While Neal (1994) and Lee et al. (2018) appeal to the central limit theorem to show equivalence, given its recent popularity in the literature (see Cressie and Johannesson, 2008a; Katzfuss, 2017), we restate the equivalence argument of Lee et al. (2018) here using a basis function representation of a GP but we defer to Lee et al. (2018) for a more rigorous treatment. For the basis function argument, recall that the Karhunen-Loève theorem states that a GP $Y(\boldsymbol{s})$ can be represented as a linear combination of basis functions as

$$Y(\boldsymbol{s}) = \lim_{P\to\infty} \sum_{p=1}^{P} e_p(\boldsymbol{s})\theta_p \tag{8}$$

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, intuitively, seen by noting that the NN, at layer L, gives a set of bases $(x_{Lp}(\boldsymbol{s}_1), \ldots, x_{Lp}(\boldsymbol{s}_n))'$ 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

$$Y(\boldsymbol{s}) = b_{L+1} + \sum_{p=1}^{P_L} x_{Lp}(\boldsymbol{s}) w_{(L+1)p} \tag{9}$$

where $w_{(L+1)p}$ are independent weights. Orthogonalizing $\{(x_{Lp}(\boldsymbol{s}_1), \ldots, x_{Lp}(\boldsymbol{s}_n))'\}$ (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 a representation of a GP.

Neal (1994) and Lee et al. (2018) 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 Chen et al. (2022) and Lee et al. (2018) in using NNs with large hidden layer widths and depths 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, 2008a), 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 used in NNs are estimated from the data rather than fixed *a priori*. Specifically, the $\boldsymbol{x}_{\ell p}(\boldsymbol{s})$ in Equation (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 learn any correlation structure present in the data whether stationary or not.

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 grid search 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 require additional computation time to fit a NN but, given the above strategies, these challenges are more surmountable than the computational challenges faced by full GPs.

3 NN Fitting Approach

3.1 NN Inputs

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 set of transformations (Trans) for both location values to create an 8×1 input layer of $(x, y, x^2, y^2, \sin(x), \sin(y), \cos(x), \cos(y))$. The last two are radial basis function expansions of the location variables in the following manner. Let $\boldsymbol{a}_i$ be the location of knot $i = 1, \ldots, A$, where A is the number of knots in the radial basis function expansion. Additionally, let $d(\boldsymbol{s}_1, \boldsymbol{s}_2) = \|\boldsymbol{s}_1 - \boldsymbol{s}_2\|$ be the Euclidean distance between locations $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ and

$$\theta = 2 \min_{i \neq j} d(\boldsymbol{a}_i, \boldsymbol{a}_j) \tag{10}$$

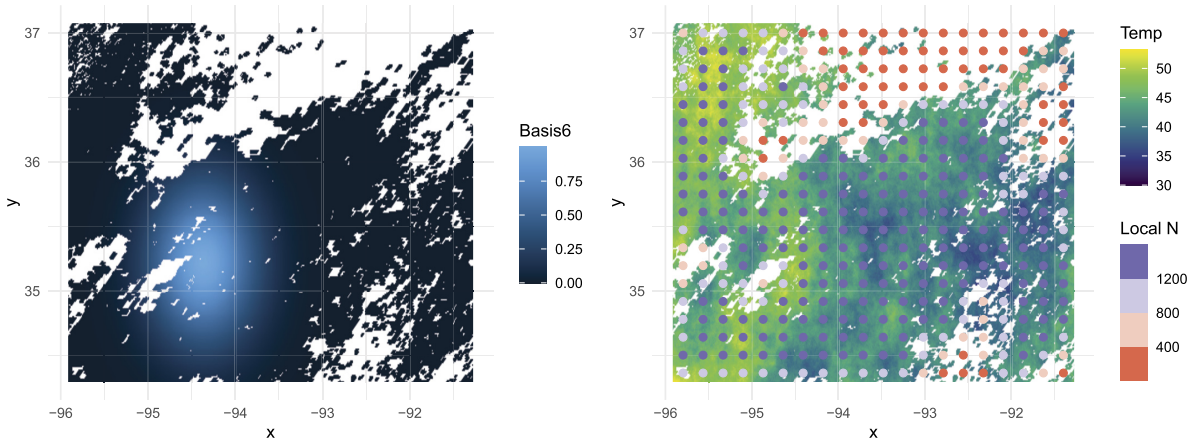
represent twice the minimum distance between basis knots. The radial basis function we used is a revision of Wendland's function for 2D data,

$$\phi(d(\boldsymbol{s}_i, \boldsymbol{a}_j)) = \mathbf{1}(d(\boldsymbol{s}_i, \boldsymbol{a}_j) \leqslant \theta) \frac{(1 - d(\boldsymbol{s}_i, \boldsymbol{a}_j)/\theta)^6 (35(d(\boldsymbol{s}_i, \boldsymbol{a}_j)/\theta)^2 + 18 d(\boldsymbol{s}_i, \boldsymbol{a}_j)/\theta + 3)}{3}. \tag{11}$$

The radial basis function expansions are, then, an element-wise transformation of $\boldsymbol{s}_i$ into $\phi(d)$. Theoretically, any basis function expansion commonly used in spatial analysis could be used here but we find radial bases to be sufficient for our purposes.

Similar to Chen et al. (2022), 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. Figure 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 locally defined and are, hence, only useful if their local sample size is large enough. That is, the radial basis functions may have near zero variance in regions with little data. Figure 1b shows the location of all four hundred knots in a radial basis function expansion to 400 areas of local correlation while the color scheme denotes the sum of the basis (i.e. $\sum_{i=1}^{N} \phi(d(\boldsymbol{s}_i, \boldsymbol{a}_j))$). Given the granularity of this



(a) (b)

Figure 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.

local basis structure, the basis-transformed variables of some basis predictors may all be 0 (i.e. near zero variance). 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.

3.2 Grid Search

For each of the four input structures investigated in model fitting, 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 a custom grid search which explores every combination of different parameter values with at most 800K weight parameters between all hidden layers. The hyperparameters contained 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 $\{0,\ 0.1\}$. This custom grid was obtained by refining an initial coarse grid of hyperparameter values done in early stages of this research (results not shown). 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. (2018) and refer to our implementation 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 framework. The following NN hyperparameters are randomly sampled for each trial: learning rate, weight decay, the weight

Table 1: The search-optimized hyperparameter settings for the best performing NNs of the custom grid search. The 1Mil-Gaus, 1Mil-NG, 150K-Sim, 150K and 1Mil-Binary datasets correspond to Examples 1–5, respectively, in Section 4.

Dataset	Input type	Hidden layers	Layer width	Epochs	Batch size	Decay rate	Dropout rate
1Mil-Gaus	MRB	4	512	60	128	1.78×10^{-6}	0
1Mil-NG	MRB	8	128	60	32	4.44×10^{-7}	0
150K-Sim	CB	4	512	20	64	0	0.1
150K	Trans	2	128	20	16	1.96×10^{-6}	0.1
1Mil-Binary	MRB	4	512	85	256	3.55×10^{-6}	0

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 $\{2^4, 2^5, \ldots, 2^8\}$. In total, 950 NNs are fit per input type using the Lee2018 grid search. Table 1 displays the optimal hyperparameter settings used in the examples given in Section 4 below.

3.3 NN Constants, Loss Functions and Stopping Criteria

The NNs across all input types and grid searches used the ReLU activation function for every layer except the output layer. Additionally, the loss function chosen for fitting NNs to continuous data was the mean squared error (MSE) (but is reported as root mean squared error (RMSE) for interpretability) while binary data minimized cross-entropy (CE). 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 (Allaire and Chollet, 2022) library in R (R Core Team, 2021) are used.

Each NN fit for hyperparameter optimization iterated through the training data a maximum of 100 epochs. After each epoch, we computed the validation loss of the updated NN. This fitting process continued until five consecutive epochs failed to achieve a new minimum validation loss. We used the minimum validation loss for each model to compare grid search model performances. For final NN model fitting with the full training set, the number of epochs trained equaled the epoch with the minimum validation loss rounded up to the nearest multiple of five.

4 Applications

In this section we give a brief 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 2 shows the *validation* performance of the best NN in each grid-input model combination. Table 3 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 which we define as the best performing model on the dataset as given by previous analyses in the literature. For Tables 2 and 3, note that *lower* RMSE and CE indicate better performance but *higher* accuracy and F1 scores indicate better performance.

Table 2: 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).

Dataset	Metric	Custom grid				Lee2018 grid			
		Raw	Trans	CB	MRB	Raw	Trans	CB	MRB
1Mil-Gaus	RMSE	0.026	0.028	0.018	**0.0078**	0.053	0.042	0.032	**0.014**
1Mil-NG	RMSE	0.151	0.133	0.112	**0.075**	0.186	0.184	0.139	**0.091**
150K-Sim	RMSE	1.27	1.26	**1.24**	1.30	1.28	1.25	**1.20**	1.25
150K	RMSE	1.85	**1.84**	1.90	1.97	1.85	**1.84**	1.92	1.98
1Mil-Binary	CE	0.163	0.149	0.096	**0.050**	0.191	0.196	0.159	**0.077**

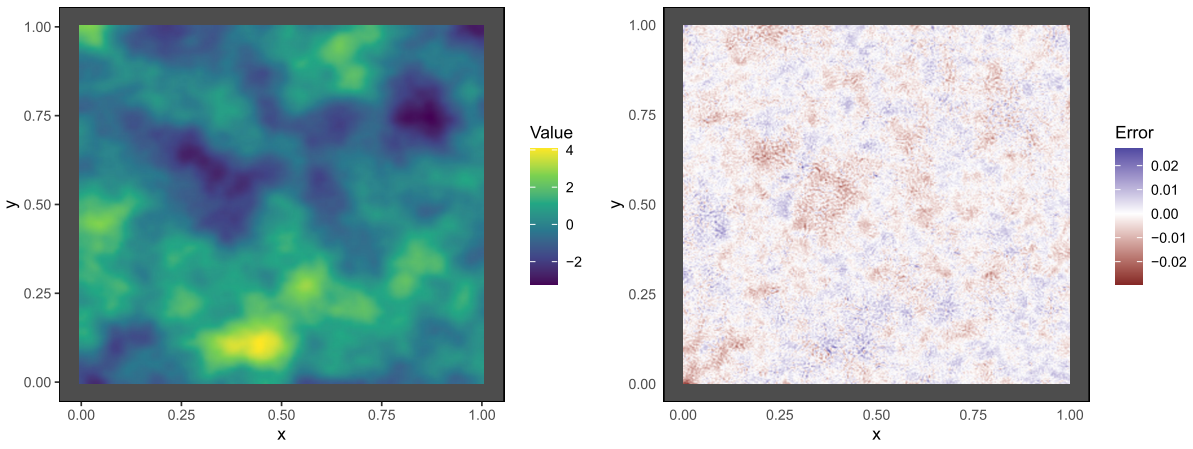
Table 3: Performance of custom and Lee2018 grid search-optimized NN setups with state-of-the-art (SoTA) for reference. Because the 1 million binary dataset is new, there is no SoTA. Only the untransformed input structure (Raw) and the basis expansion input structure with the best validation performance (be it Trans, CB, or MRB) are evaluated on the test data.

Dataset	Test metric	Custom grid		Lee2018 grid		SoTA
		Raw	Basis	Raw	Basis	
1Mil-Gaus	RMSE	0.0267	**0.0056**	0.0434	0.0130	0.00095
1Mil-NG	RMSE	0.144	**0.068**	0.160	0.105	0.021
150K-Sim	RMSE	1.27	**1.17**	1.18	1.36	0.83
150K	RMSE	**1.92**	2.49	2.85	2.36	1.53
1Mil-Binary	Accuracy	0.947	**0.986**	0.905	0.970	NA
	F1 Score	0.909	**0.976**	0.836	0.949	NA

4.1 Example 1: 1 Million Gaussian Quantitative

The first dataset considered here is of size $n = 1e6$ split into $9e5$ and $1e5$ training and testing set and is the Gaussian data from subcompetition 2b of Huang et al. (2021a). The training (Figure 2a) and test data are uniformly distributed across the x-y space, where $x, y \in (0, 1)$. For training the NN, we randomly chose 20% of the training data to be a validation set and use only the remaining 80% for training purposes but when predicting the test set, we retrain the NN using the full training data (including the validation set) using the best tuning parameters. 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.

From Table 2, under both the custom and Lee2018 grid searches, the MRB input type performed the best for the validation test set. The custom grid search outperformed Lee2018 with a validation RMSE of 0.0078 as opposed to 0.014. The performance of the MRB fit to the custom grid search also dominated among the four final models tested with a test RMSE of 0.0056 (see Table 3) and the spatial error is shown in Figure 2b. The optimal RMSE from Huang et al. (2021a) was approximately 0.001. While the NN test RMSE of 0.0056 was nearly



(a) (b)

Figure 2: (a) The 900K Gaussian training data of Example 1. (b) The average prediction error (predicted – observed) of the MRB fit with the custom grid search (the best performing model).

600% higher than the best RMSE in Huang et al. (2021a), we define a test R^2 as

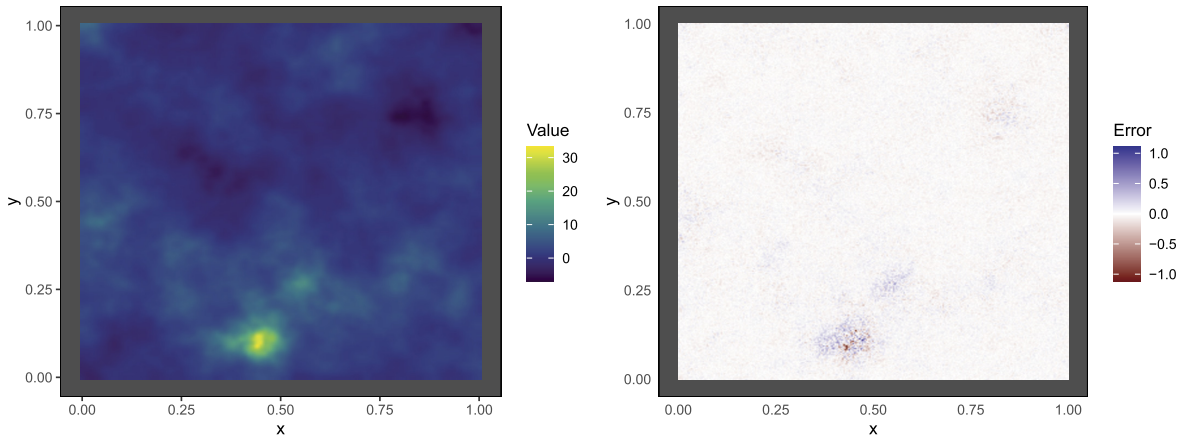
$$R^2_{\text{test}} = 1 - \frac{\sum_{i=1}^{n_{\text{test}}} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n_{\text{test}}} (y_i - \bar{y})^2},$$

which is a measure of predictive performance relative to a null model. The NN RMSE of 0.0056 equates to $R^2_{\text{test}} = 0.9999$ suggesting that NNs, while not the strongest predictor overall, can generate highly accurate predictions.

4.2 Example 2: 1 Million Non-Gaussian Quantitative

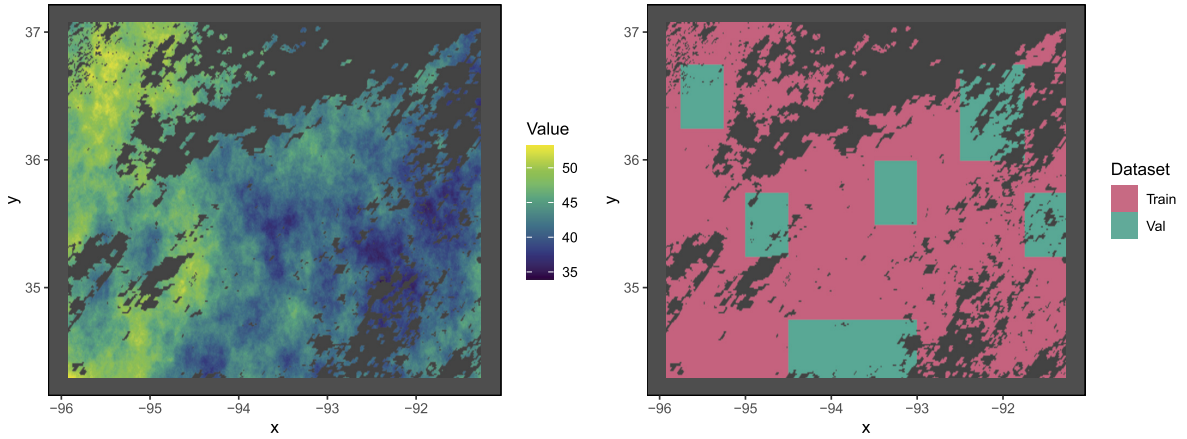
The data from this example comes from the same source as Example 1 (Huang et al., 2021a) but is a non-Gaussian dataset generated by Tukey g-and-h random fields. The training and test data are uniformly distributed across the x-y space, where $x, y \in (0, 1)$. This data was split into training and test sets of size 9×10^5 and 10^5, respectively (see Figure 3a). As in the previous example, a random sample of 20% of the training dataset was used for validation and NN training. This dataset, while similar to that of Example 1, has the added complication of a non-Gaussian error structure.

Under both the custom and Lee2018 grid searches, the MRB input type again 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 2). The MRB fit using the custom grid search also performed best with a test RMSE of 0.068. Figure 3b shows that the NN predictor had the hardest time approximating the sharp peak in the data. Overall, it overestimated how high the peak value was. Again comparing to Huang et al. (2021a), the best RMSE was 0.021 suggesting that the NN approach has a 350% higher RMSE. Again, however, the NN RMSE equates to an $R^2_{\text{test}} = 0.999$ suggesting very strong predictive performance and the difference between the best predictor from Huang et al. (2021a) and the NN is minor relative to the scale of the data.



(a) (b)

Figure 3: (a) The 9*e*5 non-Gaussian training data. (b) The average prediction error (predicted – observed) of the best performing model (the MRB fit with the custom grid search).

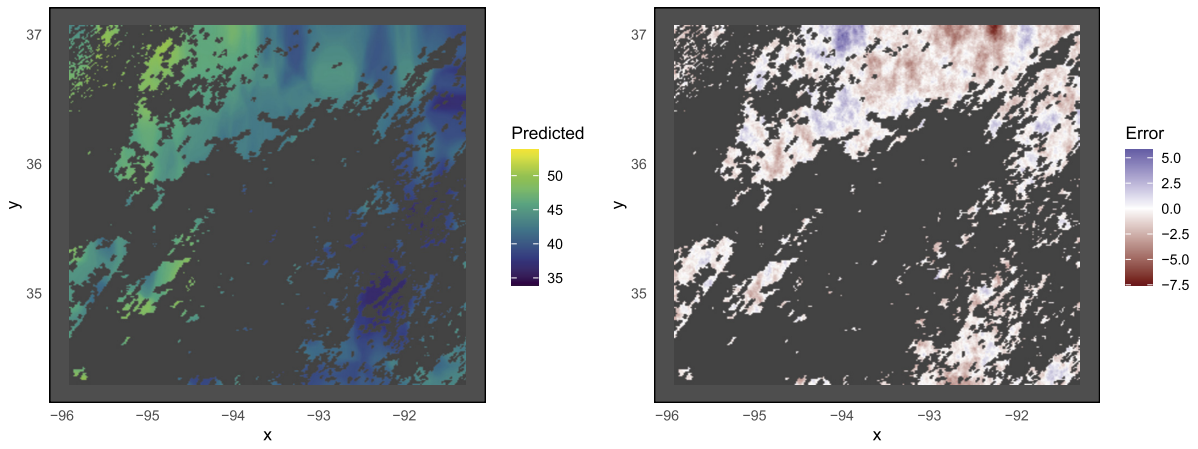


(a) (b)

Figure 4: (a) The 150K training data of Example 3. (b) Training and validation data to mimic long-range predictions during model training.

4.3 Example 3: 150K Simulated Temperatures

This dataset for this example 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). 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 4a). While this data example is substantially smaller than the previous two examples, the added complication of this example is that neither the train nor test set are distributed uniformly across the x-y coordinate space because the cloud cover blocks out groups of measurements in the location space. Hence, this dataset provides us the opportunity to evaluate NN performance relative to GP approximations in long range predictions. In an effort



(a) (b)

Figure 5: (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.

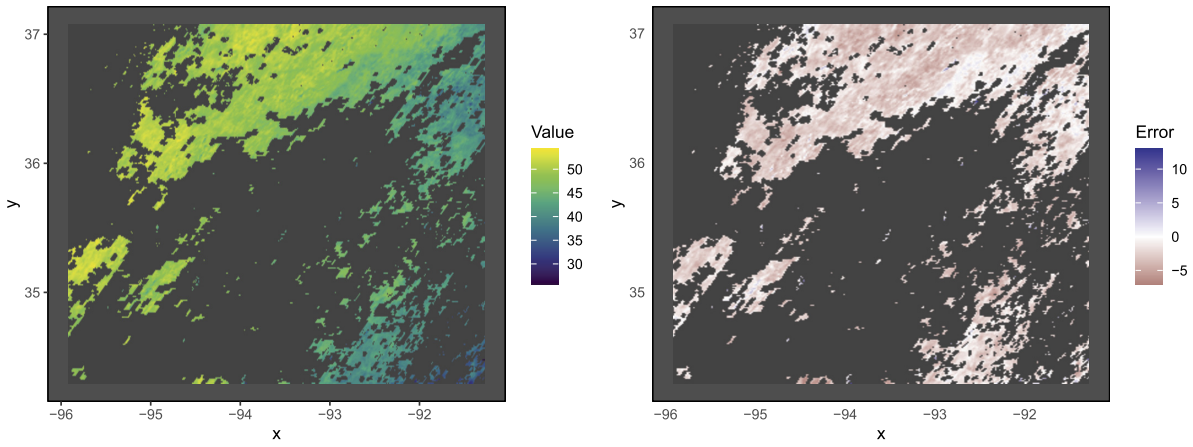
to train the data to predict in whole regions with no information, we manually divided about 20% of the data from the training set into a validation set (see Figure 4b) in a non-random fashion to validate longer range predictions in the model training.

Among all the models fit, perhaps initially surprisingly, the best model was CB (coarse basis) with a validation RMSE of 1.20. This is in contrast to the previous two examples where the MRB input models were substantially better. On the test set, the Lee2018 CB NN model performed the best with a test RMSE of 1.11. From Heaton et al. (2019), the state-of-the-art RMSE was 0.83 suggesting that the NN model had a 33% higher RMSE (see Table 3). As above, even though the RMSE for the NN was 33% higher, the $R^2_{\text{test}} = 0.858$ while the best performing model of Heaton et al. (2019) had $R^2_{\text{test}} = 0.894$ suggesting strong absolute performance of the NN model.

Looking more closely at the NN predictions, Figure 5b shows the prediction error (predicted – observed) across the locations. Notably, the NN model had the highest error for long-range predictions. This is, potentially, not surprising because long-range predictions are equivalent to extrapolation of which NNs are known to struggle (Xu et al., 2020). The long-range prediction errors also explain why the CB was preferred over the MRB; namely, the MRB did not contain enough local information for the high resolution bases to be used effectively. Hence, the NN model had to rely on coarser features to generate predictions.

4.4 Example 4: 150K Real Temperatures

This dataset is a remotely sensed daytime land surface temperature reading from the MODIS satellite on August 4, 2016 and was also included as a comparative dataset in Heaton et al. (2019). The added complication of this dataset beyond the previous examples is that the data is real, remotely-sensed temperatures and, hence, does not conform to any standard, known covariance structure. Additionally, as in the previous example, there are large areas of missing data so as to assess the ability of the NN to make long-range prediction under non-standard data structures.



(a) (b)

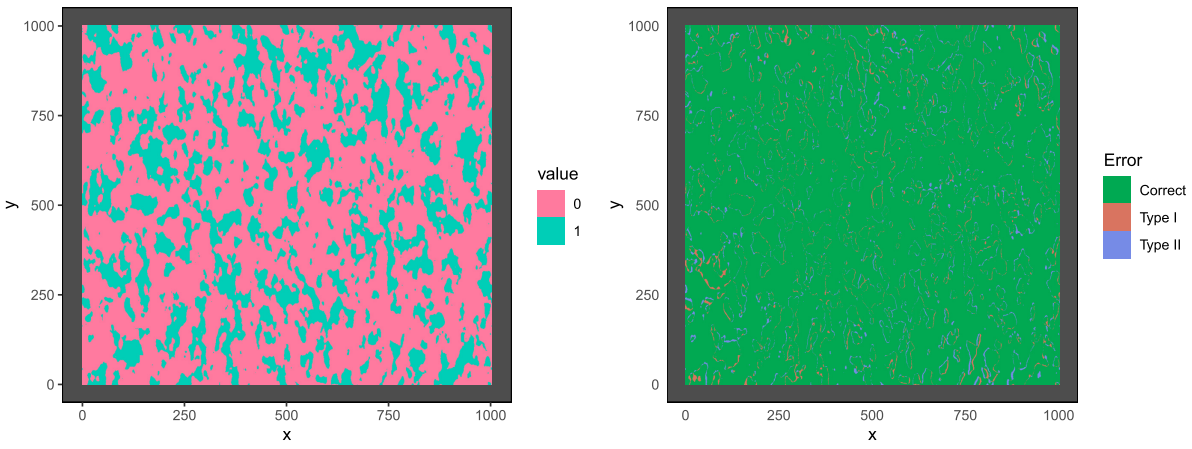
Figure 6: (a) The 50K test real temperature data. The area where only training data exist is colored gray to assist in visualizing test predictions. (b) Test prediction error for the 150K real temperature data.

Results in Table 2 again suggest that the MRB is not preferred but, rather, a more simple basis expansion (in this case the sample transformation bases) is preferred when having to do long-range predictions. As hypothesized previously, this is likely because the NN must rely on less-local information to perform prediction. Investigating the error structure of the best fit NN model in Figures 6a and 6b, the NN model seems to have larger errors for longer-range predictions. The best NN model achieved an RMSE of 1.92 (a 20% increase of the state-of-the-art methods in Heaton et al. 2019) which equates to an $R^2_{\text{test}} = 0.76$ relative to the state-of-the-art $R^2_{\text{test}} = 0.85$.

4.5 Example 5: 1 Million Binary

As a final example we consider a dataset of a binary response simulated by Sandia National Laboratories using thresholding of a Gaussian random field (see Figure 7a). An 80%–20% training-test split of the whole dataset was used to assess predictive accuracy of the NN model. Further, a random 20% of the training data was used as a validation set for NN training. This dataset represents an added challenge for the NN; namely, the ability to predict a binary spatial response. The vast majority of the GP spin-off approaches mentioned in Section 1 focus solely on quantitative (continuous) data because they do not adapt well to binary or multinomial data. NNs, on the other hand, can easily adapt to binary data by simply changing the output layer activation function.

Due to the density of the data, the MRB expansion, as was the case for Examples 1 and 2, again gives the strongest predictions with an overall test accuracy of 98.6%. Errors on the predictions from the NN are shown in Figure 7b which shows how well the NN, when provided with sufficient data, is able to reconstruct the challenging spatial structure. Specifically, only minor errors occur along transition boundaries between positive and negative responses.



(a) (b)

Figure 7: (a) The binary dataset. (b) Test prediction error for the binary dataset. A "Type I" and "Type II" error corresponds to false positive and false negative predictions.

5 Conclusions and Discussion

This research has sought to evaluate the use of neural networks for prediction given large spatial data. In the previous examples, NNs were shown to be competitive with other computationally feasible spin-offs from GPs in terms of RMSE and, in the case of binary response, accuracy or F1-score. While the NN approach did not surpass any of these alternative approaches in absolute terms, the NN predictions were quite accurate relative to the scale of the data. Hence, the main conclusion drawn from this research is that NNs are a viable approach for prediction in large spatial data problems.

While we conclude that NNs are a viable approach to large-scale prediction, we discuss a few points related to their use that should be considered: data density, computation time, hyperparameter tuning, structure design and uncertainty quantification. We consider each point in turn.

As was shown in Examples 1, 2 and 5, the NN approach can excel in situations where data is uniform and dense across the spatial domain. In contrast, as was shown in Examples 3 and 4, the NN approach struggles in situations where data is not dense (i.e. long-range predictions). Hence, when needing to perform large-scale kriging for situations where long-range predictions are necessary, we recommend using alternative approaches based on a GP which have more stable long-range prediction properties. However, additional accuracy in long-range predictions may be possible by NNs under different basis function expansions. While we used radial bases here, other possible bases include Moran bases (Hughes and Haran, 2013), bisquare bases (Cressie and Johannesson, 2008b), predictive process bases (Banerjee et al., 2008) or Wendland bases (Nychka et al., 2015). Certainly, the best choice of bases should be application and dataset specific.

Directly comparing the NN to alternative approaches in terms of computation time is challenging because computing time is highly system-specific. Relative to the computation times reported in Heaton et al. (2019), training a single neural network on Examples 3 and 4 took, approximately, 0.17 minutes (median) on the same machine as was used for that competition (and,

hence, should be comparable). Beyond the time needed to train the NN, the time to generate predictions from a trained NN model was, essentially, instantaneous even for the much larger data in Examples 1, 2 and 5. This is a strong advantage for the NN as many of the predictions for these datasets from the GP spin-offs took approximately 2500 seconds but could be as long as 20,000 seconds (as reported by Huang et al. 2021a). Further, the ability of the NN to scale with data is a strength of its use.

Of course, the time needed to train a single NN model is hardly the only time required for its implementation. By far, the most time spent for the applications above was hyperparameter tuning. That is, the time to train a single NN was multipled by the number of distinct hyperparameter settings searched over. This computation requirement can largely be parallelized so that high performance computing clusters can handle this requirement easily. For this research, we utilized 16 parallel 2.6 GHz cores on a server with 64 total core capacity and 512GB of RAM. Each core also further parallelized matrix computations to speed up training. The total computing hours needed was 169.68, 163.32, 16.89, 14.97 and 172.62 for Examples 1 through 5, respectively. Further parallelization on larger servers would speed up computations but this is what we had available to us for this research. Notably, if only a single core processor or a low-memory machine is available, the time to fully tune a NN model may be prohibitive.

Between the two grid searches used for model tuning in this paper, our custom grid search generally performed better on average than the Lee2018 grid search. 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. However, we do recognize that Bayesian methods for hyperparameter tuning are growing in popularity and may improve the results seen in this study (see Victoria and Maragatham, 2021).

Much of the hyperparameter tuning required is in relation to model structure (e.g. width and depth of the NN). Theoretically, according to Lee et al. (2018) a NN approaches a GP only as the width and/or depth approaches infinity. However, 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 8a, 8b, and 8c 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.

This paper focused solely on the use of NNs for kriging/prediction. As this is the primary use of NNs, this scope is interesting in and of itself. However, proper uncertainty quantification for the predictions may also be of interest to scientists. In such cases, the GP spin-offs discussed above may be preferred to NNs because they naturally give rise to standard errors due to underlying Gaussian assumptions. While Chen et al. (2022) described an approach to use NNs to get an uncertainty measure associated with predictions, the properties of such uncertainty measure remain largely unexplored and further research is needed in this regard.

One caveat of the NN approach compared to many others is that NNs are not specially designed to any one data problem. That is, the NN approach can easily be adapted to perform kriging for Gaussian and non-Gaussian data alike including binary, multinomial, ordered multinomial or even mixed-type spatial data. This is a strong potential advantage of the NN approach over the GP spin-offs due to its flexibility.

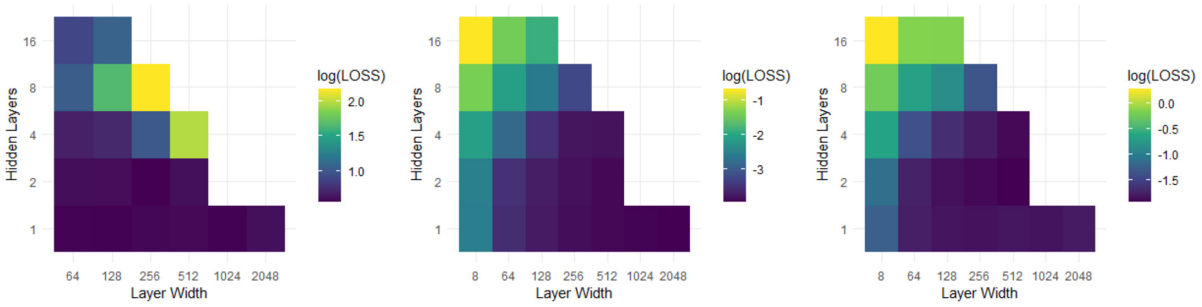


Figure 8: The best validation loss of each NN in a grid search for a given input type is aggregated in these plots. Because of right skewness, the losses were first log-transformed and then averaged across each hidden layer depth-width combination. The plots represent (a) 150K CB input, Lee2018 grid; (b) 1Mil Gaussian MRB input, custom grid; and (c) 1Mil non-Gaussian MRB input, custom grid.

Throughout this paper we have focused solely on the ability of NNs to perform large-scale spatial prediction. While NNs provide strong predictive ability as demonstrated herein, NNs are less suited for interpretability. While techniques such as variable importance are available for NNs, GP-based approaches often have a single linear effect which is easily interpretable. Hence, the decision to use NNs over GP-based approaches for spatial problems needs to account for the possible need to interpret covariate effects.

There are several other potential avenues of further research needed for using NNs in spatial data analysis. First, NN model performance needs to be considered for many other datasets, data types and data sizes. To date, NNs have only been scarcely used in the spatial statistical literature and more experience is necessary. Second, all investigated NN structures in this paper were limited to those with less than 800K parameters between the hidden layers. Investigating overparameterized models for datasets with more than 800K observations would be interesting. Third, more research is needed to determine the types of basis function expansions effective for spatial prediction. For example, the simple basis function expansion used here (Trans) could be extended to calculate additional transformations of the latitude-longitude elements. Additionally, as we saw above, the MRB approach was not effective if the data contained large areas of missing values. There may be types of basis function expansions that can work across missing value patterns. Lastly, we used the early-stop technique mentioned above to speed up grid search computations. However, that may have inhibited the performance of our grid searches in finding the best model parameterizations or not given the model enough epochs to sufficiently learn the data structure. Increasing the requirements for early stopping may yield significantly improved predictive performance in NN models.

Supplementary Material

All code and data, along with appropriate documentation, used for this research is available at https://github.com/skylerg022/nn-as-gp.

Acknowledgement

This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

Funding

This research was supported by NASA grant 80NSSC20K1594 and by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

References

Allaire J, Chollet F (2022). *keras: R Interface to 'Keras'*. R package version 2.9.0.

Banerjee S, Gelfand AE, Finley AO, Sang H (2008). Gaussian predictive process models for large spatial data sets. *Journal of the Royal Statistical Society, Series B, Statistical Methodology*, 70(4): 825–848.

Chen W, Li Y, Reich BJ, Sun Y (2022). Deepkriging: Spatially dependent deep neural networks for spatial prediction. *Statistica Sinica*. https://doi.org/10.5705/ss.202021.0277.

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

Cressie N, Johannesson G (2008b). Fixed rank Kriging for very large spatial data sets. *Journal of the Royal Statistical Society, Series B, Statistical Methodology*, 70(1): 209–226.

Cressie N, Wikle CK (2015). *Statistics for Spatio-Temporal Data*. John Wiley & Sons.

Datta A, Banerjee S, Finley AO, Gelfand AE (2016a). Hierarchical nearest-neighbor Gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, 111(514): 800–812.

Datta A, Banerjee S, Finley AO, Gelfand AE (2016b). On nearest-neighbor Gaussian process models for massive spatial data. *Wiley Interdisciplinary Reviews: Computational Statistics*, 8(5): 162–171.

Diggle PJ, Ribeiro PJ, Christensen OF (2003). An introduction to model-based geostatistics. In: *Spatial Statistics and Computational Methods*, 43–86. Springer.

Diggle PJ, Tawn JA, Moyeed RA (1998). Model-based geostatistics. *Journal of the Royal Statistical Society. Series C. Applied Statistics*, 47(3): 299–350.

El Bannany M, Khedr AM, Sreedharan M, Kanakkayil S (2021). Financial distress prediction based on multi-layer perceptron with parameter optimization. *IAENG International Journal of Computer Science*, 48: 3.

Furrer R, Genton MG, Nychka D (2006). Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics*, 15(3): 502–523.

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

Genton MG, Kleiber W (2015). Cross-covariance functions for multivariate geostatistics. *Statistical Science*, 30(2): 147–163.

Gerber F, Nychka D (2021). Fast covariance parameter estimation of spatial Gaussian process models using neural networks. *Stat*, 10(1): e382.

Heaton MJ, Datta A, Finley AO, Furrer R, Guinness J, Guhaniyogi R, et al. (2019). A case study competition among methods for analyzing large spatial data. *Journal of Agricultural, Biological, and Environmental Statistics*, 24(3): 398–425.

Higdon D (1998). A process-convolution approach to modelling temperatures in the North Atlantic Ocean. *Environmental and Ecological Statistics*, 5(2): 173–190.

Huang H, Abdulah S, Sun Y, Ltaief H, Keyes DE, Genton MG (2021a). Competition on spatial statistics for large datasets. *Journal of Agricultural, Biological, and Environmental Statistics*, 26(4): 580–595.

Huang H, Blake LR, Katzfuss M, Hammerling DM (2021b). Nonstationary spatial modeling of massive global satellite data. arXiv preprint: https://arxiv.org/abs/2111.13428.

Hughes J, Haran M (2013). Dimension reduction and alleviation of confounding for spatial generalized linear mixed models. *Journal of the Royal Statistical Society, Series B, Statistical Methodology*, 75(1): 139–159.

Jabbar H, Khan RZ (2015). Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). *Computer Science, Communication and Instrumentation Devices*, 70: 163–172.

Katzfuss M (2017). A multi-resolution approximation for massive spatial datasets. *Journal of the American Statistical Association*, 112(517): 201–214.

Katzfuss M, Guinness J (2021). A general framework for Vecchia approximations of Gaussian processes. *Statistical Science*, 36(1): 124–141.

Kaufman CG, Schervish MJ, Nychka DW (2008). Covariance tapering for likelihood-based estimation in large spatial data sets. *Journal of the American Statistical Association*, 103(484): 1545–1555.

Lee J, Sohl-dickstein J, Pennington J, Novak R, Schoenholz S, Bahri Y (2018). Deep neural networks as Gaussian processes. In: *International Conference on Learning Representations*.

Lenzi A, Bessac J, Rudi J, Stein ML (2021). Neural networks for parameter estimation in intractable models. arXiv preprint: https://arxiv.org/abs/2107.14346.

Liu H, Ong YS, Shen X, Cai J (2020). When Gaussian process meets big data: A review of scalable gps. *IEEE Transactions on Neural Networks and Learning Systems*, 31(11): 4405–4423.

Matthews A, Rowland M, Hron J, Turner RE, Ghahramani Z (2018). Gaussian process behaviour in wide deep neural networks. arXiv preprint: https://arxiv.org/abs/1804.11271.

Mesa J, Vasquez DB, Aguirre JV, Valencia JSB (2019). Sensor fusion for distance estimation under disturbance with reflective optical sensors using multi layer perceptron (mlp). *IEEE Latin America Transactions*, 17(09): 1418–1423.

Molnar C, Freiesleben T, König G, Casalicchio G, Wright MN, Bischl B (2021). Relating the partial dependence plot and permutation feature importance to the data generating process. arXiv preprint: https://arxiv.org/abs/2109.01433.

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

Nuanmeesri S, 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(5): 7714–7719.

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

Nychka D, Bandyopadhyay S, Hammerling D, Lindgren F, Sain S (2015). A multiresolution Gaussian process model for the analysis of large spatial datasets. *Journal of Computational and Graphical Statistics*, 24(2): 579–599.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Ramachandran P, Zoph B, Le QV (2017). Searching for activation functions. arXiv preprint: https://arxiv.org/abs/1710.05941.

Sang H, Huang JZ (2012). A full scale approximation of covariance functions for large spatial data sets. *Journal of the Royal Statistical Society, Series B, Statistical Methodology*, 74(1): 111–132.

Sauer A, Cooper A, Gramacy RB (2022). Vecchia-approximated deep Gaussian processes for computer experiments. arXiv preprint: https://arxiv.org/abs/2204.02904.

Sauer A, Gramacy RB, Higdon D (2022). Active learning for deep Gaussian process surrogates. *Technometrics*. https://doi.org/10.1080/00401706.2021.2008505.

Victoria AH, Maragatham G (2021). Automatic tuning of hyperparameters using Bayesian optimization. *Evolving Systems*, 12(1): 217–223.

Wikle CK, Zammit-Mangion A (2022). Statistical deep learning for spatial and spatio-temporal data. arXiv preprint: https://arxiv.org/abs/2206.02218.

Xu K, Zhang M, Li J, SS Kawarabayashi Ki D, Jegelka S (2020). How neural networks extrapolate: From feedforward to graph neural networks. arXiv preprint: https://arxiv.org/abs/2009.11848.

Yarotsky D (2018). Optimal approximation of continuous functions by very deep relu networks. In: *Conference on Learning Theory* (S Bubeck, V Perchet, P Rigollet, eds.), 639–649. PMLR.

Zammit-Mangion A, Ng TLJ, Vu Q, Filippone M (2021). Deep compositional spatial models. *Journal of the American Statistical Association*. https://doi.org/10.1080/01621459.2021.1887741.

Zammit-Mangion A, Wikle CK (2020). Deep integro-difference equation models for spatio-temporal forecasting. *Spatial Statistics*, 37: 100408.

Zhang P, Jia Y, Gao J, Song W, Leung H (2018). Short-term rainfall forecasting using multi-layer perceptron. *IEEE Transactions on Big Data*, 6(1): 93–106.