A Comparative Study of Neural Network Adaptations for Spatial Data

Bo-yu Chen [1] and Hao Zhang [2,*]

[1] Department of Statistics, Purdue University, West Lafayette, Indiana, USA
[2] Department of Statistics and Probability, Michigan State University, East Lansing, Michigan, USA
Email: chen2433@purdue.edu (B.C.); zhan2329@msu.edu (H.Z.)
*Corresponding author

***Abstract*—In this study, we evaluated the predictive performance of various deep neural network approaches for spatial data. Recent advances in neural networks have led to a surge in deep learning methods and applications. Many of these methods were developed implicitly for independent data, and it is not trivial to incorporate spatial correlation into them. However, several strategies have been proposed to adapt neural networks to account for spatial correlation. Using both simulated and real-world datasets, we compared spatial methods developed from fully connected deep neural networks, analyzing the impact of different spatial adaptations. Our findings aim to inform future research directions in this evolving field.**

***Keywords*—deep learning, Kriging, neural network, prediction, spatial statistics**

I. Introduction

Kriging is a powerful spatial interpolation method that was originally developed in the mining industry. Now widely applied in environmental sciences, hydrology, and meteorology, it utilizes variograms and correlation functions to model spatial dependencies. The method provides the best linear unbiased predictor by minimizing mean squared error, effectively bridging geostatistical analysis with advanced machine learning techniques like Gaussian process regression and kernel learning. This approach enables precise spatial estimation across diverse scientific domains, showcasing the versatility of kernel-based methodologies in pattern recognition and optimization.

Assume data are a partial realization of a stochastic process $Z(\mathrm{s}), \mathrm{s} \in D$ where D is a compact set in R^d, which can be written as:

$$Z(\mathrm{s}) = Y(\mathrm{s}) + \epsilon(\mathrm{s}) \tag{1}$$

$Y(\mathrm{s})$ is a zero mean Gaussian process with kernel $\mathrm{K}_\theta(.,.)$, and $\epsilon(\mathrm{s})$ is white noise. Suppose we have observations $\mathrm{Z} = \left(Z(\mathrm{s}_1) \ldots, Z(\mathrm{s}_n)\right)^\top$ at locations $\mathrm{s}_i \in D \subset \mathrm{R}^d, i = 1, \ldots, n$. Kriging seeks the unbiased linear predictor $\hat{Z}(s) = \sum_{i=1}^n a_i Z(s_i)$ that minimizes the mean squared error:

$$E\left(Z(s) - \hat{Z}(s)\right)^2 \tag{2}$$

The solution is given by:

$$\hat{Z}(s) = k^\top \boldsymbol{\Sigma}^{-1} \mathbf{Z} \tag{3}$$

where k is the covariance between $Z(\mathrm{s}_0)$ and $\mathbf{Z}$, $\boldsymbol{\Sigma} = \mathbf{V} + \tau^2 \mathbf{I}$ is the covariance matrix of $\mathbf{Z}$, $\mathbf{V} = \left(\mathbf{K}(s_i, s_j)\right), i, j = 1, \ldots, n$ is the variance covariance matrix of $\mathbf{Y}$, and $\mathbf{I}$ is the identity matrix. If the process is Gaussian, Kriging can be derived from the posterior distribution of a Gaussian hierarchical model. The parameters θ and τ^2 can be estimated through maximizing the likelihood function:

$$\begin{aligned} &L(\theta, \tau^2, \mu) \\ &= (2\pi)^{-\frac{n}{2}} \det(\boldsymbol{\Sigma})^{-\frac{1}{2}} \exp\left(-\frac{(\mathbf{Z}-\mu\mathbf{1})^T \boldsymbol{\Sigma}^{-1} (\mathbf{Z}-\mu\mathbf{1})}{2}\right) \end{aligned} \tag{4}$$

Additionally, Kriging is closely related to kernel ridge regression which approximates data by projecting it onto a Reproducing Kernel Hilbert Space (RKHS) [1]. According to Karhunen-Loéve expansion, a Gaussian process can be expressed as a linear combination of eigenfunctions with random coefficients:

$$Z(s) = \sum_{i=1}^n \sqrt{\lambda_i} \psi_i(s) w_i + \tau \epsilon(s) \tag{5}$$

where $\lambda_i \geqslant 0$ are eigenvalues, w_i are independent standard normal random variables, and $\psi_i(\mathrm{s})$ are eigenfunctions on D such that:

$$\int_D \psi_i(x) \psi_j(x) dx = \delta_{i,j}, \forall i, j \tag{6}$$

The above form shows that a Gaussian process can be interpreted as $Z(\mathrm{s})$ regressed on kernel features $\psi(\mathrm{s})$. Moreover, by Mercer's Theorem,

$$K(s, s') = \sum_{i=1}^n \tilde{\psi}_i(s) \tilde{\psi}_i(s') \tag{7}$$

where $\tilde{\psi}_i(s') = \sqrt{\lambda_i} \psi_j(s)$.

Given a kernel K, denote the corresponding RKHS, H_K, the kernel ridge regression minimizes the following regularized sum of least squares.

$$\sum_{i=1}^n \left(z(s_i) - f(s_i)\right)^2 + \tau^2 |f|_{H_K} \tag{8}$$

Manuscript received January 21, 2025; revised February 12, 2025; accepted March 10, 2025; published July 15, 2025.

Over all functions $f \in H_K$, where $|f|_{H_K}$ denotes the norm of f in H_K.

The closed-form solution to the above problem is [2]:

$$f^*(s) = \sum_{i=1}^{n} \alpha_i K(s, s_i) \quad (9)$$

where $(\alpha_1, \ldots, \alpha_n)^T = (\mathbf{V} + \tau^2 \mathbf{I})^{-1} \mathbf{Z}$. The solution is identical to Kriging. Therefore, Kriging prediction can be regarded as an approximation by kernel features.

A long-standing challenge in Kriging is the computation of the inverse of covariance matrix $\mathbf{\Sigma}$. Eqs. (3) and (4) above show that the matrix determinant and inversion are required for prediction and likelihood computation. It is widely recognized that the storage and computational costs for these operations on an $n \times n$ matrix are $O(n^2)$ and $O(n^3)$, respectively, which quickly become infeasible for computation as n grows.

Several approaches have been proposed to address these challenges, including local methods [3, 4], covariance tapering [5, 6], and low-rank approximations [7, 8]. Recently, a likelihood-based method based on Vecchia's approximation has gained popularity due to its computational efficiency and promising practical results [9, 10]. The key idea is to decompose the full likelihood into a product of multiple conditional likelihoods and approximate the product by conditioning only on a subset of the data:

$$p(z) = p(z_1) \prod_{i=2}^{n} p(z_i | z_{i-1}, \ldots, z_1) \approx p(z_1) \prod_{i=2}^{n} p\left(z_i \middle| z_{g(z_i)}\right) \quad (10)$$

where $g(z_i)$ is a subset of $z_1, \ldots, z_{i-1}$. Vecchia's approximation has been widely adopted in methods related to Gaussian processes, including several neural network-based approaches.

Recent advances in neural networks have caught the attention of researchers in various fields, including spatial statistics [11–14]. In addition to their outstanding predictive performance, the training and prediction processes of neural networks do not necessarily involve the computation of covariance matrices, providing an alternative approach for large spatial data prediction.

We introduce a fully connected feed-forward neural network as follows: Let the input to the neural network be denoted by s, representing the coordinates. A Deep Neural Network with L hidden layer can be specified as follows:

$$h_1(s_i) = W_0 s_i + b_0$$

$$h_2(s_i) = W_1 \phi\left(h_1(s_i)\right) + b_1$$

$$\vdots$$

$$h_L(s_i) = W_L \phi\left(h_{L-1}(s_i)\right) + b_L$$

$$f(s_i) = W_{L+1} h_{L-1}(s_i), i = 1, \ldots, n \quad (11)$$

For each layer j, h_j is the jth layer with n_j neurons, W_j, b_j are weight matrices and bias vectors, and $\phi(\cdot)$ is the activation function applied to each element of h. The model output $f(s_i)$ provides the prediction for z at location s_i. For continuous data, a commonly used loss function for training is the Ordinary Least Square (OLS) loss, or the Mean Squared Error (MSE):

$$\mathcal{L}_n(f) = \sum_{i=1}^{n} \left(f(s_i) - z(s_i)\right)^2 \quad (12)$$

Training the neural network is typically performed through back-propagation, with techniques such as Stochastic Gradient Descent (SGD) accelerating the training process. However, these methods may involve a trade-off with increased model variance.

While neural networks provide an alternative to Kriging, their direct application to spatial data often results in suboptimal performance due to the limited consideration of spatial dependence inherent in such data. A substantial body of research focuses on neural networks with spatial applications, aiming to incorporate spatial correlation into the models by altering input [15–17], adjusting model parameters [18–20], or modifying the loss function [21]. These approaches stem from perspectives such as Gaussian process regression or kernel feature learning and have been shown to outperform standard neural networks in spatial applications. However, little research has been conducted to compare the performance of these methods against one another.

In this study, we review these spatially adapted neural network methods and evaluate their predictive performance using the dataset from Heaton *et al.* [22].

II. Methods

We classify neural network adaptations for spatial data into three categories: input adaptation, parameter adaptation, and loss function adaptation.

A. Input Adaptation

One approach to incorporating spatial dependence into a neural network is to create spatial features as additional inputs. Since learning spatial dependence solely from coordinates may be inefficient and challenging, spatially dependent features such as Kriging predictions and coordinate differences from the nearest neighbors of the target location can be introduced to enhance spatial information [15]. Chen *et al.* [16] proposed DeepKriging (DK), which uses the Wendland kernel to construct a set of basis functions at different resolutions as new input for the network. Let $\{u_j\}$, $j = 1, \ldots, m$ be knots on rectangular grids, the basis functions are given by $\psi_j^*(s) = \psi(\|s - u_j\| / \beta)$, where

$$\psi(d) = \begin{cases} (1-d)^6 (35d^2 + 18d + 3)/3, & d \in [0,1] \\ 0, & \text{otherwise} \end{cases} \quad (13)$$

The compactly supported kernel is chosen to improve computational efficiency, although other positive definite functions can also be employed. When the activation function is linear, DK prediction becomes a linear combination of basis functions, which can be connected to KL-expansion as a finite number of basis functions approximates the bases of a Gaussian process with deterministic coefficients. Subsequent research has extended DK to handle bivariate and spatio-temporal data [23, 24]. Furthermore, Lin *et al.* [25] enhances the

usability and robustness of DK by incorporating multiresolution thin-plate spline functions and Huber's loss.

Wang and Zhang [26] approximates eigenvalues and eigenfunctions in the Karhunen-Loéve expansion with Nyström approach and utilizes them as features to train a deep neural network. Suppose s $\in$ [0, 1]^2 and let $\{u_{i,j}\}$ = $\{i/(m+1), j/(m+1), i, j = 1, ..., m\}$ be a set of $N = m^2$ uniformly spaced knots on [0, 1]^2. The covariance matrix of these knots is denoted by $\mathbf{\Sigma}_N$ with the ith eigenvalue and eigenvector denoted as $\lambda_{N,i}$ and $e_i = (e_{i1}, ..., e_{iN})^T$, respectively. Then, the ith eigenvalue and the corresponding eigenfunction can be approximated as follows:

$$\lambda_i = \lambda_{N,i}/N \tag{14}$$

$$\psi_i(s) = \frac{\sqrt{N}}{\lambda_{N,i}} \sum_{j=1}^{N} k(s, u_j) e_{i,j} \tag{15}$$

The number of features, p, can be chosen such that

$$\sum_{i=1}^{p} \frac{\lambda_{N,i}}{trace(\Sigma_N)} > r \tag{16}$$

where r is 0.95 or 0.99. The computation of features can be done separately and is specific to the kernel. The additional computation from obtaining the features does not make use of the kernel trick. However, once the features are obtained, training of the neural net does not involve a big covariance matrix.

In place of Wendland basis functions and eigenfunctions, the Random Ensemble Deep Spatial (REDS) model from Daw and Wikle [17] uses random Fourier features as network inputs [27]. Based on Bochner's Theorem, a continuous stationary kernel can be expressed as follows:

$$k(s,s') = \int_{R^d} \exp\left(i\omega^T(s-s')\right) p(\omega) d\omega = E_{p_\omega}[e^{(i\omega^T s)} e^{(i\omega^T s')^*}] \tag{17}$$

The superscript asterisk denotes complex conjugation. Since both $p(\omega)$ and $K(s,s')$ are real, we only need to consider the cosine part of Euler's formula. Setting $\varphi_\omega(s) = \sqrt{2}\cos(\omega^T s + b)$ where $\omega \sim p(\omega)$ and $b \sim U$ [0, 2π], the product $\varphi_\omega(s)\varphi_\omega(s')$ has an expected value K(s, s') where k is determined by $p(\omega)$. Therefore, we can approximate $K(h)$ through random projection.

For efficient computation, REDS uses the Extreme Learning Machine method, which assigns random weights to a neural network and trains only the weights in the output layer through LASSO regression. Since the inputs and most weights are random, the model simulates a large ensemble of networks to perform predictions and inferences using medians and quantiles. This approach can be implemented through parallel computing, avoiding the heavy computational burden of a single over-parameterized neural network, and provides an efficient alternative neural network approach for spatial prediction.

B. Parameter Adaptation

Another approach to adaptation is through modification of network parameters. Zammit-Mangion *et al.* [20] propose the Spatial Bayesian Neural Network, which incorporates spatially varying network parameters and a spatial embedding layer formed by radial basis functions into a Bayesian neural network. However, it is only suitable for models that are computationally efficient to simulate as the process requires generating a large number of realizations from the underlying process. Instead of treating a neural network as a direct approximation of the mapping between inputs and outputs, it can also be regarded as a feature extractor that transforms inputs from their original space to a potentially lower-dimensional feature space. Kriging can then be performed in this feature space. This approach is known as Deep Kernel Learning (DKL) [28]. DKL effectively integrates the strengths of both neural networks and Gaussian Processes. When the sample size is large, scalable Gaussian Process techniques such as Structured Kernel Interpolation can be incorporated [29]. However, it lacks clear interpretability, and without additional regularization, training can easily lead to overfitting [30, 31].

To adapt Gaussian Processes with neural networks for spatial data, Zammit-Mangion *et al.* [19] combines neural networks with the concept of spatial warping, which transforms the coordinates of a non-stationary spatial process into those of a stationary one. In their model, the warping function is constructed using either a fully connected neural network without bias terms or a deep Gaussian Process, resulting in a Spatial Input-Warped Gaussian Process (SIWGP) or a Spatial Deep Stochastic Process (SDSP). Both models produce injective mappings, ensuring that the spatial relationships between data points remain intact after warping. The formulation of the model is described as follows:

$$Z(s_i) = Y(s_i) + \epsilon_i, i = 1, \dots, n$$

$$Y(s_i) = W_L \psi\left(f(s)\right)$$

$$f(s) = g_L \circ \cdots \circ g_1(s)$$

$$g_i(s) = W_i \phi_i(s) \tag{18}$$

where ψ is a set of bisquare basis functions. In each layer, g is referred to as the "warping function," encompassing an activation function, rescaling and weights multiplication. They implemented different choices of warping units, including the Axial Warping Units which warp inputs from the previous layer either linearly or nonlinearly using combinations of non-negative weights and a sigmoid function, the RBF Units which use radial basis functions for local contraction and expansion, and Mobius Transformation Units which considers the real and imaginary components. Treating f either randomly or deterministically can result in a SDSP or a SIWGP. In this study, we focus on SIWGP, as it is estimated using maximum likelihood estimation and can be regarded as an extension of DKL to spatial data. The model has been further extended to multivariate and separable spatio-temporal cases in [18, 32].

C. Loss Function Adaptation

Training a neural network with MSE (OLS loss) can be regarded as estimating the mean function of a Gaussian process with covariance matrix $\mathbf{\Sigma}= \mathbf{I}$. Therefore, it does not account for the spatial covariance between locations. Assuming additional covariates x_i are observed at each location s_i, we have the following model:

$$Z(s_i) = f(x_i) + Y(s_i) + \epsilon(s_i) = f(x_i) + \tilde{\varepsilon}(s_i) \quad (19)$$

The first term is a deterministic mean function that can be estimated using neural networks, and the second term is a Gaussian Process. Similar to linear regression, a Generalized Least Square (GLS) loss (20) can be considered for lower variance. This is the NN-GLS model proposed by Zhan and Datta [21].

$$\mathcal{L}_n^g(\boldsymbol{f}) = \frac{1}{n}\big(\boldsymbol{Z} - \boldsymbol{f}(x)\big)^T \boldsymbol{Q}_\theta (\boldsymbol{Z} - \boldsymbol{f}(x)) \quad (20)$$

$\boldsymbol{Q}_\theta = \mathbf{\Sigma}_\theta^{-1}$ is the precision matrix and is estimated through Vecchia's approximation. NN-GLS iteratively updates θ and network parameters until they converge. Then, predictions can be made through Kriging with Vecchia's approximation and a mean function estimated by the neural network.

The method can be regarded as a special type of graph neural network with decorrelation layers. Fitting a neural network with GLS loss is equivalent to fitting a neural network with additional decorrelation layers using OLS loss. Therefore, the computation of NN-GLS, including mini-batching, back-propagation, and bootstrapping, can be done similarly to vanilla neural networks. Theoretical results including the finite sample error rate and the existence of the sieve estimator and asymptotic consistency are also proved in the paper.

The original NN-GLS separates the mean function and spatial random effect into two components, making it inapplicable to spatial data without covariates. In this study, we extend the approach in Wang and Zhang [26] by combining eigenfeatures generated in Eq. (15) with NN-GLS for prediction. The original process can be expressed as:

$$Z(s) = \sum_{i=1}^{p} w_i \psi_i(s) + \sum_{p+1}^{\infty} w_i \psi_i(s) + \epsilon(s) \quad (21)$$

Eigenfeatures approximate the first term in the right-hand side by neural networks as a truncated KL-expansion, while the remaining term constitutes is another zero mean spatially-correlated Gaussian process. Thus, by taking $\hat{\psi}_j(s_i)$, j = 1, …, p from Eq. (15) as covariates, we could use NN-GLS to model spatial data without covariates.

III. Comparison

We evaluated the performance of neural networks and their spatial adaptations for prediction tasks using Root Mean Square Error (RMSE). Six methods were selected for comparison: the vanilla Neural Network (NN), DKL, DK, REDS, SIWGP, and NN-GLS.

SIWGP was implemented using the function `deepspat_nn_GP` in the `R` package `deepspat`, extended by Vu *et al.* [32] to incorporate Vecchia's approximation for efficient computation with large datasets. DKL was implemented using the Python package `GPytorch`, and NN-GLS was implemented with Python package `geospaNN`. The rest of the methods were implemented with `Pytorch`. The analysis was conducted on the Gilbreth cluster at Purdue University, leveraging A100 GPUs.

To ensure computational feasibility, the hyperparameter tuning process involves selecting a subset of 10,000 points from the training dataset, which is then split into an 80% training set and a 20% validation set for optimization. The neural network hyperparameters were tuned manually and with the Python package `Optuna`, with candidate depths ranging from 1 to 20, nodes from 20, 40, …, 300, and batch sizes from 50, 100, 150. For DK and NN-GLS, additional tuning was performed with nodes due to their high feature dimensionality. The networks were trained for 200 epochs using the Adam optimizer with an initial learning rate of 0.1, which was reduced based on the validation loss [33]. Activation functions were selected between a sigmoid function and a Rectified Linear Unit (ReLU), depending on performance. To prevent overfitting, dropout layers, which randomly deactivate nodes, and an early stopping rule, which halts training when the validation loss ceases to decrease, were applied.

All kernel-related methods-DKL, REDS, SIWGP, and NN-GLS use the exponential kernel for both simulated and real datasets. Following Heaton *et al.* [22], the likelihood-based approach assumes a constant mean in simulated data and a linear effect for latitude and longitude in satellite data. In DKL, the grid size for kernel approximation was optimized using `Optuna`, with values ranging from 50, …, 2,000. For DK, following the recommendations in [16], the scale parameter β is set to be 0.5. While the resolution levels were suggested based on the sample size, we restricted the levels to four due to computational constraints, resulting in 7,159 basis inputs.

For REDS, hyperparameters were taken from the original paper, with 500 ensembles created for prediction. In NN-GLS, we constructed eigenfeatures based on Eq. (15). Covariance parameters were estimated using 2,500 randomly sampled locations with maximum likelihood ahead, resulting in 1,640 features for simulated data and 4,651 features for satellite data.

A. Data

We selected the datasets studied in Heaton *et al.* [22], which include a simulated dataset and a real dataset measured by the Moderate Resolution Imaging Spectroradiometer (MODIS) instrument aboard the Terra satellite. Both datasets share the same 500×300 grid, spanning longitude values from –95.91153 to –91.28381 and latitude values from 34.29519 to 37.06811.

The MODIS dataset includes daytime land surface temperatures recorded on August 4, 2016. After removing corrupted data due to cloud cover, 148,309 observations were retained for the study, with 105,569 used for training and 42,740 for testing.

The simulated dataset consists of 150,000 points generated from a Gaussian process with an exponential covariance kernel. The parameters-range, marginal variance, nugget, and the mean were set to (4/3, 16.40, 0.05, 44.49), and estimated from a random sample of 2,500 observations from the MODIS data. The training set contains 105,569 points, and the test set includes 44,431 points.

Both datasets are displayed in Fig. 1.

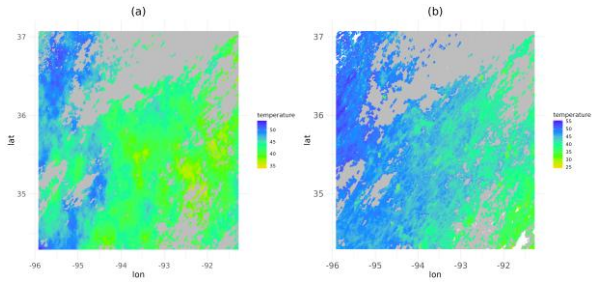

Fig. 1. Training data from (a) Simulated data (b) Satellite data. Gray points are test locations.

B. Discussion

The results are shown in Table I. Overall, SIWGP demonstrates the best performance, followed by REDS, DK, and NN-GLS. NN is slightly inferior, with DKL ranking last. The poor performance of DKL may be attributed to its corresponding objective function, the log Gaussian likelihood. As noted by Ober *et al.* [30], maximum likelihood estimation with over-parameterized models, such as neural networks, is prone to overfitting. This occurs because the data-fit term, $-(Z-\boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(Z-\boldsymbol{\mu})$, has a maximum, whereas the model complexity term, $-\log|\boldsymbol{\Sigma}|$, does not. Consequently, when the model has sufficient degrees of freedom, it may attempt to make the covariance matrix singular, leading to overfitting. This phenomenon, where output features overlap, is referred to as feature collapse in [22]. The pathological behavior can be mitigated by regularizing the model during training. For example, Liu *et al.* [34] imposes a bi-Lipschitz constraint on a feature extractor to ensure the model remains sensitive to input variations. Based on this approach, Amersfoort *et al.* [31] combines the bi-Lipschitz constraint with the inducing point method for Gaussian likelihood estimation, enhancing uncertainty quantification. On the other hand, the injective restriction in SIWGP prevents overlapping spatial locations and avoids singular covariance matrices, achieving significant improvements despite using the likelihood function as its objective. Additionally, the bisquare function in the top layer provides a flexible structure for spatial modeling.

The performance of DK is better than NN in both simulated and real data, demonstrating the effectiveness of embedding inputs with basis functions. In our analysis, there was one instance where NN achieved an RMSE significantly lower than DK; however, this result was not reproducible in subsequent runs. Since all spatial information originates from the coordinates, this suggests that NN has the potential to achieve performance similar to DK and other methods with properly tuned parameters. However, such instances are rare and unpredictable. Typically, a one-layer NN performs sufficiently well, whereas DK requires more layers for optimal performance, although its performance tends to decline when the number of layers approaches 10 or more.

By design, NN-GLS should have a greater capability to model spatial correlation than DK. While both methods use low-rank approximations, NN-GLS incorporates additional covariance through the GLS loss. However, our results show that DK performs better on simulated data. A possible explanation is that the parameters estimated from a subset of points are not sufficiently accurate, causing the eigenfeatures to underperform Wendland bases. An interesting application would be to combine DK with NN-GLS for further improvement. Additionally, NN-GLS uses an exponential kernel to decorrelate the second term in Eq. (21), which corresponds to an unknown kernel. When applied to a dataset simulated from an exponential kernel, the issue of misspecification arises, leading to inferior performance.

The success of REDS likely stems from several factors, including its ensemble approach, use of random bases, and regularization through LASSO regression. The use of multiple ensembles allows REDS to mitigate the impact of extreme predictions from random samples. The random Fourier transform provides spatial embedding for the coordinate inputs, enabling the model to effectively capture spatial variability. Additionally, LASSO regression in the final layer facilitates feature selection and helps prevent overfitting. Although we used the correct kernel for implementation, our results for simulated data were worse, while those for satellite data were better than the results reported in Daw and Wikle [17]. As a finite rank approximation, REDS does not perform as well as SIWGP in simulated data, but it outperforms other methods.

Table II lists the computation time for each method. Although direct comparisons may not be entirely fair, as some methods were implemented using GPU and others using CPU, all methods completed estimation and prediction within a reasonable amount of time, avoiding the infeasible matrix computations required by Kriging. With the fewest input features, NN demonstrated the shortest computation time for satellite data among all methods. The higher computation time for simulated data is likely a random anomaly, though the computation still completed within two minutes. Similarly, DK is efficient in estimation once the features are generated. NN-GLS, on the other hand, takes more time as the package does not support GPU computation and the method must update the precision matrix for the loss function during the process. The similar computation times observed for DKL and DK across both datasets are due not only to the well-developed kernel approximation but also to DKL's tendency to overfit. As noted earlier, DKL is prone to overfitting, leading to early termination through the early stopping rule. SIWGP requires significant computation time for warping and basis function calculations. Additionally, its implementation using `R` and `TensorFlow` may contribute to increased runtime. The

runtime of REDS, on the other hand, depends on the number of available cores for parallel computing. On our cluster, REDS achieved a favorable balance between computation time and accuracy. Overall, the input adaptation approaches can be integrated with GPU or parallel computation conveniently, whereas other approaches depend on the support of the package. Kriging-based approaches, while providing more accurate results, require more computation time when the sample size scales up.

For further comparison, we include prediction results and computation times from Kriging methods used in Heaton *et al.* [22], as shown in Tables I and II. These methods include Nearest Neighbor Gaussian Process (NNGP), Fixed Rank Kriging (FRK), Stochastic Partial Differential Equations (SPDE), Local Approximate Gaussian Processes (LAGP), and Tapering. Each method corresponds to a different approximation of Gaussian processes: Vecchia's approximation (NNGP), low-rank methods (FRK), sparse precision matrices (SPDE), local approaches (LAGP), and sparse covariance matrices (Tapering).

In their results, SPDE achieves the best performance, while NNGP has a better balance between computation time and accuracy. For data generated from a Gaussian process, parametric methods outperform all neural network methods except SIWGP, as they both correctly specify the model structure. For satellite data, while SPDE and NNGP still achieve the lowest RMSE, neural network approaches such as SIWGP and NN-GLS demonstrate comparable performance

TABLE I. RMSE FOR EACH METHOD AND FROM HEATON *ET AL*. [22]

Method	Simulated	Satellite
NN	1.51	2.31
DKL	2.04	4.47
DK	1.09	2.19
REDS	1.24	1.89
SIWGP	0.86	1.69
NN-GLS	1.46	1.79
NNGP	0.88	1.64
FRK	1.31	2.44
SPDE	0.86	1.53
LAGP	1.11	2.08
Tapering	0.97	2.45

TABLE II. COMPUTATION TIME (MIN), INCLUDING METHODS FROM HEATON *ET AL*. [22]

Method	Simulated	Satellite
NN	1.53	0.292
DKL	0.56	0.559
DK	0.32	0.536
REDS	15.75	16.87
SIWGP	167.04	199.88
NN-GLS	24.80	75.07
NNGP	1.99	2.06
FRK	2.18	2.32
SPDE	138.34	120.33
LAGP	2.28	2.27
Tapering	188.36	133.26

IV. CONCLUSION

In this study, we compare the prediction RMSE of six fully connected neural network-based methods using datasets from Heaton *et al*. The result demonstrates that incorporating spatial correlation into neural networks significantly improves predictive performance.

Among all methods, a Kriging-based approach in parameter adaptation, SIWGP, has the best predictive performance. While input adaptation methods are less accurate, they can be easily implemented through GPU or parallel computing to provide much lower computation time. NN-GLS in loss function adaptation is less accurate than SIWGP, but its further improvement may be a future research direction.

We also compare Kriging approximations introduced in Heaton *et al.* Their performance is competitive with neural networks in terms of both accuracy and computational speed, demonstrating the reliability of traditional approaches in spatial data prediction.

Instead of spatial data, studying dependent data with high-dimensional input could be another intriguing topic. It is well known that kernel-based methods suffer from the curse of dimensionality, whereas neural networks can overcome this limitation [35, 36]. On the other hand, most spatial adaptation methods discussed in this study require constructing basis functions, random features, or kernels. The input dimension not only affects kernel performance but also determines the number of required features, impacting computational speed. Therefore, a detailed comparison between spatially adapted neural networks presents a potential direction for further study.

Finally, Kriging approximations and neural networks with parameter adaptation or loss function adaptation naturally provide uncertainty quantification under the Gaussian process assumption. Input adaptation methods, on the other hand, require additional approaches such as bootstrapping or conformal prediction to quantify uncertainty, which may further increase computation time. An exception is the REDS method, which generates a set of ensembles that can be used for uncertainty quantification. Future work includes comparing the uncertainty quantification performance of different methods.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

B.C. was responsible for data analysis and original draft preparation; H.Z. contributed to supervision, review and editing; all authors approved the final version of the manuscript.

REFERENCES

[1] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, The MIT Press, 2005. https://doi.org/10.7551/mitpress/3206.001. 0001

[2] G. Wahba, "Spline models for observational data," in *CBMS-NSF Regional Conference Series in Applied Mathematics*, Society for Industrial and Applied Mathematics, 1990. https://doi.org/10.1137/1.9781611970128

[3] R. B. Gramacy and D. W. Apley, "Local Gaussian process approximation for large computer experiments," *Journal of*

Computational and Graphical Statistics, vol. 24, no. 2, pp. 561–578, April 2015. https:// doi.org/10.1080/10618600.2014.914442

[4] C. Park and D. Apley, "Patchwork Kriging for large-scale Gaussian process regression," *J. Mach. Learn. Res.*, vol. 19, no. 1, 269–311, Jan. 2018.

[5] R. Furrer, M. G. Genton, and D. Nychka, "Covariance tapering for interpolation of large spatial datasets," *Journal of Computational and Graphical Statistics*, vol. 15, no. 3, pp. 502–523, Sept. 2006. https: //doi.org/10.1198/106186006X132178

[6] C. G. Kaufman, M. J. Schervish, and D. W. Nychka, "Covariance tapering for likelihood-based estimation in large spatial data sets," *J. Amer. Statist. Assoc*., vol. 103, no. 484, pp. 1545–1555, 2008.

[7] N. Cressie and G. Johannesson, "Fixed rank kriging for very large spatial data sets," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 70, no. 1, pp. 209–226, 2008. https://doi.org/10.1111/j.14679868.2007.00633.x

[8] M. Lázaro-Gredilla, J. Quiñonero-Candela, C. E. Rasmussen, and A. R Figueiras-Vidal, "Sparse spectrum gaussian process regression," *Journal of Machine Learning Research*, vol. 11, no. 63, pp. 1865–1881, 2010.

[9] A. Datta, S. Banerjee, A. O. Finley, and A. E. Gelfand, "Hierarchical nearest-neighbor Gaussian process models for large geostatistical datasets," *J. Amer. Statist. Assoc*., vol. 111, no. 514, pp. 800–812, 2016. https://doi.org/10.1080/01621459.2015.1044091

[10] M. Katzfuss and J. Guinness, "A general framework for Vecchia approximations of gaussian processes," *Statist. Sci*., vol. 36, no. 1, pp. 124–141, 2021. https://doi.org/10.1214/19-STS755

[11] C. K Wikle and A. Zammit-Mangion, "Statistical deep learning for spatial and spatiotemporal data," *Annual Review of Statistics and Its Application*, vol. 10, no. 1, pp. 247–270, 2023.

[12] S. Wang, J. Cao, and P. S. Yu, "Deep learning for Spatio temporal data mining: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 8, pp. 3681–3700, 2022.

[13] P. Tziachris, V. Aschonitis, T. Chatzistathis, M. Papadopoulou, and I. (John) D. Doukas, "Comparing machine learning models and hybrid geostatistical methods using environmental and soil covariates for soil pH prediction," *International Journal of Geo-Information*, vol. 9, no. 4, 276, 2020. https://doi.org/10.3390/ijgi9040276

[14] P. Du, X. Bai, K. Tan, Z. Xue, A. Samat, J. Xia, E. Li, H. Su, and W. Liu, "Advances of four machine learning methods for spatial data handling: A review," *Journal of Geovisualization and Spatial Analysis*, vol. 4, no. 1, 13, 2020. https://doi.org/10.1007/s41651-020-00048-5

[15] H. Wang, Y Guan, and B. Reich, "Nearest-neighbor neural networks for Geostatistics," in *Proc. 2019 International Conference on Data Mining Workshops (ICDMW)*, IEEE, Beijing, China, 2019, pp. 196–205. https://doi.org/10.1109/ICDMW.2019.00038

[16] W. Chen, Y. Li, B. Reich, and Y. Sun, "DeepKriging: Spatially dependent deep neural networks for spatial prediction," *Statistica Sinica*, vol. 34, no. 1, pp. 291–311, 2024.

[17] R. Daw and C. K. Wikle, "REDS: Random ensemble deep spatial prediction," *Environmetrics*, vol. 34, no. 1, e2780, 2023. https://doi.org/10.1002/env.2780

[18] Q. Vu, A. Zammit-Mangion, and N. Cressie, "Modeling nonstationary and asymmetric multivariate spatial covariances via deformations," *Statistica Sinica*, vol. 32, no. 4, pp. 2071–2093, 2022.

[19] A. Zammit-Mangion, T. L. J. Ng, Q. Vu, and M. Filippone, "Deep compositional spatial models," *J. Amer. Statist. Assoc.*, 2021. https://doi.org/10.1080/01621459.2021.1887741

[20] A. Zammit-Mangion, M. D. Kaminski, B. Tran, M. Filippone, and N. Cressie, "Spatial Bayesian neural networks," arXiv preprint, arXiv:2311.09491, 2023.

[21] W. Zhan and A. Datta, "Neural networks for geospatial data," *J. Amer. Statist. Assoc*., 2024. https://doi.org/10.1080/01621459.2024.2356293

[22] M. J. Heaton, A. Datta, A. O. Finley, R. Furrer, J. Guinness, R. Guhaniyogi, F. Gerber, R. B. Gramacy, D. Hammerling, M. Katzfuss, F. Lindgren, D. W. Nychka, F. Sun, and A. Zammit-Mangion, "A case study competition among methods for analyzing large spatial data," *Journal of Agricultural, Biological and Environmental Statistics*, vol. 24, no. 3, pp. 398–425, 2019. https://doi.org/10.1007/s13253-018-00348-w

[23] P. Nag, Y. Sun, and B. J. Reich, "Bivariate DeepKriging for largescale spatial interpolation of wind fields," arXiv preprint, arXiv:2307.08038, 2023.

[24] P. Nag, Y. Sun, and B. J. Reich, "Spatio-temporal DeepKriging for interpolation and probabilistic forecasting," arXiv preprint, arXiv:2306.114722023.

[25] D. Lin, H. Huang, and S. Tzeng, "Some enhancements to DeepKriging," *Stat*., vol. 12, no. 1, e559, 2023. https://doi.org/10.1002/sta4.559

[26] S. Wang and H. Zhang, "Deep learning and kriging," in *Proc. 2024 7th World Conference on Computing and Communication Technologies (WCCCT)*, IEEE, Chengdu, China, 2024, pp. 309–313. https://doi.org/10.1109/WCCCT60665.2024.10541416

[27] A. Rahimi, and B. Recht, "Random features for large-scale kernel machines," *Advances in Neural Information Processing Systems*, 2007.

[28] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep kernel learning," in *Proc. the 19th International Conference on Artificial Intelligence and Statistics*, Cadiz, Spain, 2016, pp. 370–378.

[29] A. Wilson and H. Nickisch, "Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP)," in *Proc. the 32nd International Conference on Machine Learning*, Lille France, 2015, pp. 1775–1784.

[30] S. W. Ober, C. E. Rasmussen, and M. van der Wilk, "The promises and pitfalls of deep kernel learning," in *Proc. the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, 2021, pp. 1206–1216.

[31] J. van Amersfoort, L. Smith, A. Jesson, O. Key, and Y. Gal, "On feature collapse and deep kernel learning for single forward pass uncertainty," arXiv preprint, arXiv:2102.11409, 2022.

[32] Q. Vu, A. Zammit-Mangion, and S. J. Chuter, "Constructing large nonstationary spatio-temporal covariance models via compositional warpings," *Spatial Statistics*, vol. 54, 100742, 2023. https://doi.org/10.1016/j.spasta.2023. 100742

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd International Conference on Learning Representations, ICLR 2015*, San Diego, CA, USA, 2015.

[34] J. Zhe Liu, Z. Lin, S. Padhy, D. Tran, T. Bedrax-Weiss, and B. Lakshminarayanan, "Simple and principled uncertainty estimation with deterministic deep learning via distance awareness," in *Proc. the 34th International Conference on Neural Information Processing Systems (Vancouver, BC, Canada) (NIPS' 20)*, 2020.

[35] B. Ghorbani, S. Mei, T. Misiakiewicz, and A. Montanari, "When do neural networks outperform kernel methods?" *Advances in Neural Information Processing Systems*, vol. 33, pp. 14820–14830, 2020.

[36] M. Refinetti, S. Goldt, F. Krzakala, and L. Zdeborová, "Classifying high-dimensional Gaussian mixtures: Where kernel methods fail and neural networks succeed," in *Proc. International Conference on Machine Learning*, 2021, pp. 8936–8947.