Journal of Data Science 0 (0), 1–15
??? 2025
DOI: 10.6339/25-JDS1210
Statistical Data Science

A Scalable Spatial Decorrelation Preprocessing Approach for Machine and Deep Learning

Matthew J. Heaton[1,*], Andrew Millane[1], and Jake S. Rhodes[1]
[1]*Department of Statistics, Brigham Young University, Provo, UT 84602, USA*

Abstract

Spatial data display correlation between observations collected at nearby locations. Generally, machine and deep learning methods either do not account for this correlation or do so indirectly through correlated features. To account for spatial correlation, we propose preprocessing the data using a spatial decorrelation transform motivated from properties of a multivariate Gaussian distribution and Vecchia approximations. The preprocessed, transformed data can then be ported into a machine or deep learning tool. After model fitting on the transformed data, the output can be spatially re-correlated via the corresponding inverse transformation. We show that including this spatial adjustment results in higher predictive accuracy on simulated and real spatial datasets.

Keywords *Gaussian process; predictive accuracy; Vecchia approximation; whitening transformation*

1 Research Synopsis and Goals

"Spatial data," broadly speaking, refers to data that contain geographic or location-based information about the observations. Such data arise across a wide range of scientific disciplines, including ecology (Plant, 2018), environmental science (Harris and Jarvis, 2014; Yuan et al., 2020), public health (Waller and Gotway, 2004; Shaddick and Zidek, 2015), real estate (Pace et al., 1998; Arbia et al., 2021), and civil engineering (Ziakopoulos and Yannis, 2020). A core challenge in analyzing spatial data is accounting for the inherent correlation among observations collected at nearby locations. Spatial statistics—the branch of statistics dedicated to developing methods that model and utilize this dependence (Stein and Gelfand, 2022)—has long relied on the Gaussian process (GP; Gelfand and Schliep 2016) as a flexible and principled framework for spatial modeling. However, GPs scale poorly with data size due to $\mathcal{O}(n^3)$ covariance matrix operations (Bradley et al., 2016; Heaton et al., 2019; Huang et al., 2021), and their flexibility can be limited by simplifying assumptions such as linearity (Banerjee et al., 2014).

Because of these limitations of the GP, recent attention has naturally turned to machine and deep learning (ML/DL) methods for spatial prediction, which can model nonlinear effects and handle large data sets efficiently. Standard machine and deep learning models generally assume that individual training samples are independent, an assumption violated when data exhibit spatial dependence leading to biased results and suboptimal predictions. Consequently, considerable research has focused on adapting ML/DL methods to appropriately account for spatial correlation. Approaches such as random forests (Nikparvar and Thill, 2021; Patelli et al.,

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

Received June 13, 2025; Accepted November 25, 2025

2024), fully connected neural networks (Wikle and Zammit-Mangion, 2023), convolutional neural networks (Zammit-Mangion et al., 2022), graph neural networks (Sainsbury-Dale et al., 2025; Cisneros et al., 2024; Tonks et al., 2024), and deep Gaussian processes (Vu et al., 2022; Sauer et al., 2023) have all been adapted for spatial settings. Most of these adaptations incorporate spatial information through feature engineering—e.g., including coordinates, basis functions, or neighborhood structures as inputs to the model (Patelli et al., 2024; Sekulić et al., 2020; Georganos et al., 2021; Gray et al., 2022; Lin et al., 2023; Chen et al., 2024; Bradley et al., 2011; Zammit-Mangion et al., 2024). Yet, spatial feature engineering can oversmooth data and increase dimensionality, which may obscure signal and leave residual spatial correlation (Stein, 2014; Bishop, 2006). Other adaptation efforts modify the loss function itself to incorporate spatial covariance structures (Saha et al., 2023; Zhan and Datta, 2025), though such methods can become computationally demanding unless approximations such as Vecchia methods (Katzfuss and Guinness, 2021) or high-performance implementations (Abdulah et al., 2018) are used.

In this work, we propose a complementary strategy that is both computationally efficient and broadly compatible with existing ML/DL frameworks. We introduce a *spatial whitening transformation*—a preprocessing step motivated by properties of multivariate Gaussian distributions and Vecchia approximations—that removes local spatial correlation by adjusting each observation based on its nearest neighbors under a parametric correlation model. The decorrelated data can then be analyzed using any standard ML/DL model and loss function, and predictions from the fitted model are back-transformed to recover the appropriate spatial dependence. The method is computationally scalable, avoids feature inflation, and integrates seamlessly with standard machine-learning workflows without modifying model architectures.

Section 2 details the methodology and mathematical foundation; Section 3 evaluates its predictive performance through simulations; Section 4 applies it to a real-world $PM_{2.5}$ dataset; and Section 5 concludes with directions for future extensions to non-Gaussian and more complex spatial structures.

2 Methodology

2.1 Background Under Gaussian Assumption

Let $\boldsymbol{Y} = (Y(\boldsymbol{\ell}_1), \ldots, Y(\boldsymbol{\ell}_n))'$ be a response variable measured at the locations $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_n$, where $\boldsymbol{\ell}_i \in \mathcal{D} \subset \mathbb{R}^d$, $\mathcal{D}$ is the spatial domain, and d is an integer (typically $d = 2$ in spatial statistics, such that $\boldsymbol{\ell}_i = (\ell_{i1}, \ell_{i2})'$ would correspond to longitude and latitude coordinates). Further, let $\boldsymbol{x}(\boldsymbol{\ell}) = (x_0(\boldsymbol{\ell}), \ldots, x_P(\boldsymbol{\ell}))'$ be a vector of features (covariates or inputs), where we assume that $x_0(\boldsymbol{\ell}) \equiv 1$ is an intercept term. As motivation for our methods, for this section we assume (this assumption will be dropped in subsequent sections) that $Y(\boldsymbol{\ell})$ follows a GP with linear mean $\mu(\boldsymbol{\ell}) = \boldsymbol{x}'(\boldsymbol{\ell})\boldsymbol{\beta}$, where $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_P)'$ is a vector of coefficients such that

$$\boldsymbol{Y} \sim \mathcal{N}_n(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2 \boldsymbol{R}), \tag{2.1}$$

where $\mathcal{N}_n$ is the n-dimensional Gaussian distribution, $\boldsymbol{X}$ is the $n \times (P+1)$ design matrix of features (with a leading column of 1's), σ^2 is a common variance (also referred to as the sill in spatial statistics terminology), and $\boldsymbol{R} = \{\rho_{ij}\}$ is the $n \times n$ spatial correlation matrix with $\rho_{ij} = \rho(d_{ij} \mid \boldsymbol{\phi})$ being a parametric correlation function of distance between observations i and j (d_{ij}) with parameters $\boldsymbol{\phi}$ (e.g., nugget, range, and smoothness parameters). For the derivations that follow, we assume a well-behaved covariance structure such that the correlation matrix

$\boldsymbol{R}$ is positive definite and invertible, which is satisfied under standard choices of stationary correlation functions and an increasing-domain asymptotic framework. These assumptions are typical in Gaussian process settings (Katzfuss and Guinness, 2021).

The factorization of the $n \times n$ matrix $\boldsymbol{R}$, where n is the number of observations, makes model (2.1) computationally infeasible for even moderately sized data sets. To develop a more computationally feasible approach for spatial data that can subsequently be extended to machine and deep learning models, recall that any multivariate probability density function (PDF) can be factored as a series of univariate conditional distributions. In our consideration, the multivariate Gaussian distribution in (2.1) can be equivalently factored as a series of univariate Gaussian conditional distributions, such that

$$\mathcal{N}_n(\boldsymbol{Y} \mid \boldsymbol{X}\boldsymbol{\beta}, \sigma^2\boldsymbol{R}) \equiv \mathcal{N}_1(Y(\boldsymbol{\ell}_1) \mid \mu_1, \sigma^2) \prod_{i=2}^{n} \mathcal{N}_1(Y(\boldsymbol{\ell}_i) \mid \mu_i, \sigma^2 v_i), \tag{2.2}$$

where $\mathcal{N}_1(y \mid m, v)$ is the univariate Gaussian distribution for y with mean m and variance v. While not explicitly shown in (2.2), the distributions in the product are conditional distributions on previous data points $\{Y(\boldsymbol{\ell}_1), \ldots, Y(\boldsymbol{\ell}_{i-1})\}$ through the mean and variance terms, which are

$$\mu_i = \begin{cases} \boldsymbol{x}'(\boldsymbol{s}_1)\boldsymbol{\beta}, & \text{if } i = 1, \\ \boldsymbol{x}'(\boldsymbol{s}_i)\boldsymbol{\beta} + \boldsymbol{R}(i, \mathcal{L}_i)\boldsymbol{R}^{-1}(\mathcal{L}_i, \mathcal{L}_i)(\boldsymbol{Y}_{\mathcal{L}_i} - \boldsymbol{X}_{\mathcal{L}_i}\boldsymbol{\beta}), & \text{if } i > 1, \end{cases} \tag{2.3}$$

$$v_i = \begin{cases} 1, & \text{if } i = 1, \\ 1 - \boldsymbol{R}(i, \mathcal{L}_i)\boldsymbol{R}^{-1}(\mathcal{L}_i, \mathcal{L}_i)\boldsymbol{R}(\mathcal{L}_i, i), & \text{if } i > 1, \end{cases} \tag{2.4}$$

where $\mathcal{L}_i = \{1, \ldots, i-1\}$ is the set of indices of preceding points, $\boldsymbol{Y}_{\mathcal{A}}$ is the set of $Y(\boldsymbol{\ell}_i)$ corresponding to indices $i \in \mathcal{A}$, $\boldsymbol{X}_{\mathcal{A}}$ are the rows of $\boldsymbol{X}$ corresponding to $\mathcal{A}$, and $\boldsymbol{R}(\mathcal{A}, \mathcal{B})$ is the submatrix of $\boldsymbol{R}$ with rows indexed by $\mathcal{A}$ and columns by $\mathcal{B}$.

Importantly, the factorization in (2.2) is still computationally demanding because the forms for μ_i and v_i in (2.3) and (2.4) require dealing with large matrices through $\boldsymbol{R}(\mathcal{L}_i, \mathcal{L}_i)$, since the cardinality of $\mathcal{L}_i$ grows with i. For computational feasibility, we adopt a nearest-neighbor approach based on the Vecchia process approximation framework (see Datta et al., 2016a,b; Katzfuss and Guinness, 2021), replacing $\mathcal{L}_i$ with a conditioning set $\mathcal{C}_i$ defined as the set of, at most, C nearest neighbors of $\boldsymbol{\ell}_i$ in terms of Euclidean distance. In general, the conditioning sets $\mathcal{C}_i$ define neighborhoods under a graphical structure determined by the correlation function. Under the stationary correlation structure adopted here, these graph-based neighbors coincide with the nearest points in Euclidean distance, hence we use the term "nearest neighbors" throughout. This restriction keeps $\boldsymbol{R}(\mathcal{C}_i, \mathcal{C}_i)$ at most $C \times C$, making computations manageable while retaining the essential local spatial dependence.

The Vecchia approximation is closely related to composite likelihood (CL) approaches in that both approximate the full likelihood using products of lower-dimensional conditional densities (Lindsay et al., 2011). The key distinction is that CL yields a pseudo-likelihood that does not define a proper joint distribution, whereas the Vecchia factorization defines a valid joint model under its assumed conditional independence structure. In this sense, Vecchia's method combines the computational scalability of composite likelihood with the generative validity of a full probabilistic model.

In using the Vecchia approximation, we must consider (i) the choice of C and (ii) the ordering of the observations. Early investigations by Datta et al. (2016a) found that $C \approx 20$ was typically sufficient for isotropic spatial processes, while Katzfuss and Guinness (2021) showed that $C \approx 10$

often suffices. In a small simulation comparing neighborhood sizes $C \in \{5, 10, 20, 30, 40\}$, predictive RMSE in the whitened space remained nearly constant (3.61–3.63) while computation time increased modestly with C. Accordingly, we selected $C = 30$ as a conservative yet efficient choice. Regarding ordering, Katzfuss and Guinness (2021) discuss the impact of observation ordering on the Vecchia approximation and recommend certain orderings to obtain better approximations; for this research, we follow their max–min suggested ordering (see also Guinness, 2018).

Under the factorization in (2.2), consider the transformation,

$$\widetilde{Y}(\boldsymbol{\ell}_i) = \begin{cases} Y(\boldsymbol{\ell}_i), & \text{if } i = 1, \\ v_i^{-1/2}\left(Y(\boldsymbol{\ell}_i) - \boldsymbol{R}(i, \mathcal{C}_i)\boldsymbol{R}^{-1}(\mathcal{C}_i, \mathcal{C}_i)\boldsymbol{Y}_{\mathcal{C}_i}\right), & \text{if } i > 1, \end{cases} \tag{2.5}$$

where $\widetilde{Y}(\boldsymbol{\ell}_i)$ denotes the spatially whitened response. Under this transformation, $\widetilde{Y}(\boldsymbol{\ell}_i)$ will also be Gaussian with expectation

$$\mathbb{E}[\widetilde{Y}(\boldsymbol{\ell}_i) \mid \boldsymbol{Y}_{\mathcal{L}_i}] = \widetilde{\boldsymbol{x}}_i'(\boldsymbol{\ell}_i)\boldsymbol{\beta}, \tag{2.6}$$

and common variance σ^2, where

$$\widetilde{\boldsymbol{x}}'(\boldsymbol{\ell}_i) = \begin{cases} \boldsymbol{x}'(\boldsymbol{\ell}_i), & \text{if } i = 1, \\ v_i^{-1/2}\left(\boldsymbol{x}'(\boldsymbol{\ell}_i) - \boldsymbol{R}(i, \mathcal{C}_i)\boldsymbol{R}^{-1}(\mathcal{C}_i, \mathcal{C}_i)\boldsymbol{X}_{\mathcal{C}_i}\right), & \text{if } i > 1. \end{cases} \tag{2.7}$$

Specifically, under (2.5), $\widetilde{Y}(\boldsymbol{\ell}_i) \overset{ind}{\sim} \mathcal{N}(\widetilde{\boldsymbol{x}}_i'(\boldsymbol{\ell}_i)\boldsymbol{\beta}, \sigma^2)$, or equivalently, $\widetilde{\boldsymbol{Y}} \sim \mathcal{N}(\widetilde{\boldsymbol{X}}\boldsymbol{\beta}, \sigma^2\boldsymbol{I})$, where $\widetilde{\boldsymbol{Y}} = (\widetilde{Y}(\boldsymbol{\ell}_1), \ldots, \widetilde{Y}(\boldsymbol{\ell}_n))'$ and $\boldsymbol{I}$ is an identity matrix. Notably, under the Gaussian assumption, transformation (2.5) serves as a spatial whitening transformation that removes local correlation.

2.2 Proposed Spatial Adjustment for Machine and Deep Learning

Given the desirable properties of the transformations of $Y(\boldsymbol{\ell}_i)$ and $\boldsymbol{x}'(\boldsymbol{\ell}_i)$ in (2.5) and (2.7) under the Gaussian case, we propose using these transformations as a general decorrelation step prior to machine and deep learning model fitting. Importantly, in ML and DL applications we do not assume that $\boldsymbol{Y}$ follows a Gaussian distribution; rather, we apply (2.5) to spatial data to mitigate the effects of spatial correlation during model training. Although the whitening transformation is motivated by Gaussian process theory, it does not require Y to be normally distributed. It can therefore be used with general continuous, quantitative outcomes.

The transformation of $Y(\boldsymbol{\ell}_i)$ to $\widetilde{Y}(\boldsymbol{\ell}_i)$ via (2.5) has several important implications for ML and DL methods. First, because (2.5) removes the influence of neighboring observations $\boldsymbol{Y}_{\mathcal{C}_i}$ on $Y(\boldsymbol{\ell}_i)$, methods that rely on data resampling (e.g., random forests) or minibatching (e.g., neural networks) can be applied directly without regard to spatial structure. When data are spatially correlated, subsets of $\boldsymbol{Y}$ cannot typically be used because spatial dependence affects the entire vector (see Saha et al., 2023). In contrast, subsets or bootstrap samples of the whitened data $\widetilde{\boldsymbol{Y}}$ can be drawn freely due to their approximate independence. The whitening transformation therefore enables resampling strategies analogous to block bootstrap approaches: once spatial dependence has been removed, standard resampling or minibatching procedures used in ML/DL can be applied safely without violating independence assumptions among samples.

Second, the transformed matrix of feature variables (inputs) $\widetilde{\boldsymbol{X}}$ forms a linear basis for $\widetilde{\boldsymbol{Y}}$ if the data are Gaussian. Hence, when using this transformation to move beyond linearity in ML and DL approaches, we propose using $\widetilde{\boldsymbol{x}}(\boldsymbol{\ell})$ from (2.7) as the appropriate input features into

Algorithm 1 Spatial adjustment for machine and deep learning.

Choose spatial tuning parameters $\boldsymbol{\phi}$.
Choose machine-specific tuning parameters $\boldsymbol{\theta}$.
Order observations according to the max-min criterion of Katzfuss and Guinness (2021).
Determine nearest neighbor sets $\mathcal{C}_i$ for $i = 1, \ldots, n$.
Transform $Y(\boldsymbol{\ell}_i)$ to $\widetilde{Y}(\boldsymbol{\ell}_i)$ via (2.5).
Transform $\boldsymbol{x}(\boldsymbol{\ell}_i)$ to $\widetilde{\boldsymbol{x}}(\boldsymbol{\ell}_i)$ via (2.7).
Fit machine or deep learning model $\widehat{f}(\widetilde{\boldsymbol{x}}(\boldsymbol{\ell}) \mid \boldsymbol{\theta})$ to transformed data $\{(\widetilde{Y}(\boldsymbol{\ell}_i), \widetilde{\boldsymbol{x}}(\boldsymbol{\ell}_i))\}$.
for each prediction location $\boldsymbol{u} \in \mathcal{D}$ **do**
    Determine nearest neighbor set of $\boldsymbol{u}$ ($\mathcal{C}_{\boldsymbol{u}}$) among training locations $\{\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_n\}$
    Transform $\boldsymbol{x}(\boldsymbol{u})$ to $\widetilde{\boldsymbol{x}}(\boldsymbol{u})$ via (2.7)
    Obtain a prediction $\widetilde{Y}^\star(\boldsymbol{u}) = \widehat{f}(\widetilde{\boldsymbol{x}}(\boldsymbol{u}) \mid \boldsymbol{\theta})$.
    Inverse transform the prediction $\widetilde{Y}^\star(\boldsymbol{u})$ to original scale prediction $Y^\star(\boldsymbol{u})$ according to (2.8).
    Output prediction $Y^\star(\boldsymbol{u})$.
end for

the ML and DL function rather than the original $\boldsymbol{x}(\boldsymbol{\ell})$. Importantly, because the untransformed $\boldsymbol{x}(\boldsymbol{\ell}_i)$ *includes* an intercept term (i.e., $x_0(\boldsymbol{\ell}) \equiv 1$), after the transformation in (2.7), $\widetilde{\boldsymbol{x}}(\boldsymbol{\ell})$ includes a transformed intercept term. This transformed intercept term is important because the conditioning set indexed by $\mathcal{C}_i$ varies with observation with the first observation being a marginal distribution as per Equation (2.2). Keeping the intercept term in the transformation of $\boldsymbol{x}(\boldsymbol{\ell}_i)$ to $\widetilde{\boldsymbol{x}}(\boldsymbol{\ell}_i)$ accounts for this variability in the conditioning set and is needed for our approach in spite of intercepts not being traditionally included as features in machine or deep learning models.

Third, scaling by $v_i^{-1/2}$ gives that all the $\widetilde{Y}(\boldsymbol{\ell}_i)$ have a common variance σ^2. Under a common variance, common loss functions such as squared error loss are appropriate (rather than, e.g., weighted least squares). Hence, any machine or deep learning model can be fit in the standard way according to any chosen loss function.

After training the machine learning model on the transformed data $\{\widetilde{Y}(\boldsymbol{\ell}_i), \widetilde{\boldsymbol{x}}(\boldsymbol{\ell}_i)\}_{i=1}^n$, predictions for a new response, $Y(\boldsymbol{u})$, at an unobserved location $\boldsymbol{u} \in \mathcal{D}$ is obtained as follows. First, let $\mathcal{C}_u \subset \{1, \ldots, n\}$ denote the set of indices corresponding to the C nearest neighbors to $\boldsymbol{u}$ from among the training set locations $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_n$. Given $\mathcal{C}_u$, define $\widetilde{\boldsymbol{x}}(\boldsymbol{u})$ as in (2.7) then input $\widetilde{\boldsymbol{x}}(\boldsymbol{u})$ into the fitted machine learning model to get a prediction $\widetilde{Y}^\star(\boldsymbol{u}) = \widehat{f}(\widetilde{\boldsymbol{x}}(\boldsymbol{u}))$ where $\widehat{f}$ is the trained model. We can then back transform the prediction $\widetilde{Y}^\star(\boldsymbol{u})$ to a prediction for $Y(\boldsymbol{u})$ (i.e., recorrelate the prediction with its spatial neighbors) via the inverse transformation

$$Y^\star(\boldsymbol{u}) = v^{1/2}\widetilde{Y}^\star(\boldsymbol{u}) + \boldsymbol{R}(\boldsymbol{u}, \mathcal{C}_u)\boldsymbol{R}^{-1}(\mathcal{C}_u, \mathcal{C}_u)\boldsymbol{Y}_{\mathcal{C}_u}, \tag{2.8}$$

where we use the notation $\boldsymbol{R}(\boldsymbol{u}, \mathcal{C}_u)$ to denote $\mathrm{Corr}(Y(\boldsymbol{u}), \boldsymbol{Y}_{\mathcal{C}_u})$ under the correlation function $\rho(\cdot)$.

In brief, our proposed method is as follows. First, inputs and outputs are transformed according to (2.5) and (2.7). Second, the machine or deep learning model of choice is fit to the transformed data $\{(\widetilde{Y}(\boldsymbol{\ell}_i), \widetilde{\boldsymbol{x}}(\boldsymbol{\ell}_i))\}_{i=1}^n$. Third, the model predicts the transformed $\widetilde{Y}^\star(\boldsymbol{u}) = \widehat{f}(\widetilde{\boldsymbol{x}}(\boldsymbol{u}))$ and the output is inverse-transformed via (2.8). This algorithm is detailed as Algorithm 1.

2.3 Notes on Implementation

The above adjustment for spatial correlation in the data can be used for, in theory, for any machine or deep learning model of choice. However, there are a few details that are important to consider in the implementation which we discuss here. First, our proposed spatial adjustment is inherently different from Saha et al. (2023) and Zhan and Datta (2025) in that their approaches use a generalized squared error loss function $(\boldsymbol{Y} - \boldsymbol{f}(\boldsymbol{X}))'\boldsymbol{R}^{-1}(\boldsymbol{Y} - \boldsymbol{f}(\boldsymbol{X}))$ which is *not* the same as the independent squared error loss function $(\widetilde{\boldsymbol{Y}} - \boldsymbol{f}(\widetilde{\boldsymbol{X}}))'(\widetilde{\boldsymbol{Y}} - \boldsymbol{f}(\widetilde{\boldsymbol{X}}))$ used here because the correlation matrix $\boldsymbol{R}$ does not commute into the function $f(\cdot)$ except in linear cases. Consequently, the function $f(\cdot)$ learned under the two loss functions will *not* be the same function. Generally, this is not an issue as the interpretation of $f(\cdot)$ is not of direct interest in machine and deep learning.

In practice, the transformation in (2.5) requires specifying the parameters of the spatial correlation function $\rho(\cdot)$, in addition to any tuning parameters of the chosen machine or deep learning model (e.g., number of trees in a random forest). In traditional statistical modeling, spatial correlation parameters are often estimated from the data. However, this is problematic in our context because the residual spatial correlation structure depends on the form of the mean function. A nonlinear model may explain different aspects of the data than a linear one, leading to differing patterns of residual spatial dependence. Since our approach performs decorrelation before model fitting, we treat the spatial correlation parameters as tuning parameters, selected to complement the model type being used.

Most commonly used correlation functions have parameters with well-defined and bounded domains. For example, as shown by Berrett and Calder (2016), nugget effects can be constrained to the interval $[0, 1]$. Likewise, rescaling the spatial coordinates to $[0, 1] \times [0, 1]$ allows spatial range parameters to be bounded. This bounding makes it feasible to tune spatial parameters via grid search (e.g., Finley et al., 2019) or with more adaptive approaches like Bayesian optimization (Wu et al., 2019; Turner et al., 2021). This tuning process effectively tailors the spatial transformation to the characteristics of the downstream predictive model.

In terms of computational complexity, the whitening transformation based on the Vecchia approximation reduces the cost of spatial modeling from the $\mathcal{O}(n^3)$ operations required for a full Gaussian-process covariance factorization to $\mathcal{O}(nC^3)$, where $C = \max_i |\mathcal{C}_i|$ is the fixed neighborhood size. Because C is typically small (e.g., $C = 30$ in our experiments), the procedure scales linearly with n in both computation and memory. This makes the transformation practical for very large spatial datasets. For reference, the whitening step for $n = 50{,}000$ points required about 90 seconds on a standard workstation, after which ML/DL fitting times were comparable to those for independent data (see Table 1).

3 Simulation Studies

3.1 Setup

In this section, we evaluate the performance of our spatial adjustment using three simulated scenarios of increasing complexity. In the first scenario, the relationship between the features $\boldsymbol{X}$ and the response is linear, with no spatial correlation. This allows us to assess whether the decorrelation transform in (2.5) behaves appropriately when spatial correlation is absent—that is, whether it can be tuned to effectively recognize and adapt to an independence scenario. The second scenario introduces spatial correlation into the same linear model, allowing us to

examine the benefits of accounting for spatial dependence in a simple predictive setting. The third scenario is the most complex, combining a nonlinear relationship between features and response with spatially correlated data. For each scenario, we generate 50 simulated datasets of size $n = 50{,}000$, with spatial locations $\boldsymbol{\ell}_i$ sampled uniformly over the unit square for $i = 1, \ldots, n$. Each dataset is split into 40,000 training observations and 10,000 test observations.

Detailing the first scenario, the response is simulated according to a standard linear regression model where $\boldsymbol{Y} \sim \mathcal{N}_n(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2\boldsymbol{I})$ where $\boldsymbol{X}$ is a $n \times 10$ matrix of features where elements were simulated independently from a standard Gaussian distribution. We drew the true $\boldsymbol{\beta}$ parameters independently from $\mathcal{N}(0, 5^2)$ distribution and then randomly selected J of the 10 covariates to have zero effect where J followed a binomial distribution with 10 trials (corresponding to each of the 10 covariates) and success probability 0.5. Thus, the complete set of features contained features that did not relate to the response. Finally, we set $\sigma^2 = 100$, which roughly corresponds to an expected coefficient of determination of 0.5.

The second simulated scenario is similar to the first other than we let $\boldsymbol{Y} \sim \mathcal{N}_n(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2\boldsymbol{R})$ where $\boldsymbol{R}$ is a spatial correlation matrix constructed using the correlation function

$$\rho_{ij} = \begin{cases} 1 & \text{if } i = j, \\ (1-\omega)\exp\left(\|\boldsymbol{\ell}_i - \boldsymbol{\ell}_j\|/\phi\right) & \text{if } i \neq j, \end{cases} \tag{3.1}$$

where $\omega \in [0, 1]$ is a nugget term and $\phi > 0$ is a range parameter (see Banerjee et al., 2014, Chapter 2 for details). Specifically, we use $\phi = 0.236$ and a nugget $\omega = 0.25$, which corresponds to a spatial range (i.e., the distance at which the correlation decays to 0.05) of $\sqrt{2}/2$ (half of the maximum spatial distance on the spatial domain).

Finally, in the third scenario we simulate $\boldsymbol{Y} \sim \mathcal{N}_n(\boldsymbol{f}(\boldsymbol{X}), \sigma^2\boldsymbol{R})$ where $\boldsymbol{f}(\boldsymbol{X})$ is a non-linear function simulated from a zero-mean Gaussian process in x_1 and x_2. That is $\boldsymbol{f} \sim \mathcal{N}_n(\boldsymbol{0}, \sigma_f^2\boldsymbol{M})$ where $\boldsymbol{M}$ is determined by a stationary Matern correlation function in x_1 and x_2 with smoothness 2.1, range 0.842 and no nugget (the value of ϕ was chosen so that the spatial range was half of the maximum distance). By simulating $\boldsymbol{f}$ in this manner, the resulting relationship between the response and features is non-linear.

We employ linear models (LM), Bayesian additive regression trees (BART), single-layer perceptrons (SLP), boosting, random forests (RF), and K nearest neighbors (KNN) both with and without the spatial adjustment proposed in (2.5). To tune the above algorithms, we use a grid search across 5 values of each tuning parameter and finalized tuning parameters chosen to minimize the 5-fold cross-validation root mean square error (RMSE). For BART, we consider only the number of trees as a tuning parameter with grid values of {50, 100, 150, 200, 250}. For SLP, we tune the number of hidden neurons ({3, 5, 7, 9, 11}), penalty parameters on the weights ({1×10^{-10}, 3.16×10^{-8}, 1×10^{-5}, 3.16×10^{-3}, 1}), and the number of epochs ({10, 257, 505, 752, 1000}) while we fix the activations as ReLU (except the output layer which is linear) with no dropout. For boosting, we tune the tree depth ({1, 4, 8, 11, 15}), number of trees ({1, 500, 1000, 1500, 2000}), and learning rate ({1×10^{-10}, 1.77×10^{-8}, 3.16×10^{-6}, 5.62×10^{-4}, 0.1}). For RF, we tune the number of variables at each split ({1, 3, 5, 7, 9}) and minimum number of observations per leaf ({2, 11, 21, 30, 40}) but we fix the number of trees at 500. Finally, for KNN, we tune the number of neighbors ({1, 3, 5, 7, 10}). Additionally, when implementing the spatial decorrelation transform we tuned the nugget (ω) and range (ϕ) parameters in (3.1) using grids of {0, 0.2475, 0.4950, 0.7425, 0.99} and {0, 0.001, 0.004, 0.087, 2.040}, respectively.

We compare our spatial adjustment approach to using the spatial basis functions as inputs as advocated by, among others, Georganos et al. (2021), Gray et al. (2022), Lin et al. (2023),

Table 1: Comparison of predictive (out-of-sample) RMSE for independent, spatial basis-function, and spatial decorrelation approaches across three simulation scenarios. Bold numbers indicate the lowest RMSE within each scenario. As expected, the linear model (LM) achieved the best performance in the linear scenarios, whereas the nonlinear BART model was best in the nonlinear scenario. The proposed spatial adjustment improves RMSE for the spatial scenarios while leaving performance unchanged in the case of spatial independence and consistently outperforming the spatial basis-function approach.

	Independent Linear			Spatial Linear			Spatial Non-Linear		
Model	Ind	Basis	Decorr	Ind	Basis	Decorr	Ind	Basis	Decorr
LM	**9.97**	9.97	9.97	8.66	6.42	**5.51**	10.31	8.24	7.47
RF	10.24	10.60	10.24	8.98	7.54	6.01	9.06	6.27	5.79
SLP	10.01	10.56	10.05	8.67	7.10	5.68	9.23	6.74	6.11
Boost	10.27	10.72	10.27	8.96	7.71	6.05	9.04	6.67	5.79
BART	10.02	10.04	10.03	8.73	5.68	5.57	9.01	5.79	**5.77**
KNN	11.94	11.89	10.73	10.69	8.76	7.07	10.82	7.57	6.55

Chen et al. (2024), and Zammit-Mangion et al. (2024). For comparison, we use two resolutions of bisquare basis functions. For the coarse resolution, we use 9 bases with knot locations chosen according to a Latin Hypercube Design (LHD) and cutoff distance (scale) of 0.5 which corresponds to 1.5 times the maximum-minimum distance between knot locations. For the finer resolution, we use 25 bases with knot locations again chosen using an LHD and cutoff distance (scale) of 0.3 (1.5 times the maximum-minimum distance between fine resolution knot locations). These bases added 34 input dimensions beyond the 10 input dimensions of $\boldsymbol{X}$.

Ideally, we would have also compared the spatial random forests of Saha et al. (2023) as well as the spatial neural network of Zhan and Datta (2025). However, implementing these methods on a single simulated dataset took over 24 hours for the spatial random forest and 8.6 hours for the spatial neural networks. Hence, tuning these methods was not computationally feasible (e.g. the spatial neural network model across the 125 possible tuning parameter sets would have taken approximately 1000 hours per dataset). Although both the spatial random forest and spatial neural network could in principle be parallelized across multiple computing nodes, their total computational demand would still be several orders of magnitude greater than that of the proposed whitening transformation. Our method requires roughly 90 seconds for $n = 50{,}000$ observations and scales linearly with data size, making it practical on a single workstation.

3.2 Results

Table 1 displays the median RMSE across the 50 simulated datasets for each method in each of the scenarios. All approaches exhibit similar performance for Scenario 1 (linear, independent). This equivalence is expected because of the simplicity of the simulation scenario. However, these results are important to consider because they indicate that the proposed spatial decorrelation method does not hinder the prediction performance when there is no spatial correlation in the data. The spatial transformation can be tuned to account for the apparent lack of spatial correlation in the data.

Under Scenario 2 (linear, spatial), spatial versions of each of the algorithms (both basis and

decorrelation) outperform the independent version. However, the spatial decorrelation approach also had a lower RMSE than the basis function approach. On average, the spatial decorrelation approach reduced the RMSE by 34.4% from the independent approach and 16.2% from the basis function approach. These results suggest that both the basis function and decorrelation approaches are useful for spatial data. We hypothesize that the further improvement of the decorrelation approach over the basis function approach might relate to the increased dimensionality from the basis functions.

In Scenario 3 (non-linear, spatial), spatial models demonstrate superior performance compared to assuming independence with the decorrelation approach again realizing a larger reduction in RMSE. In this scenario, on average, the spatial decorrelation approach reduced the RMSE by 34.7% from the independent approach and 8.7% from the basis function approach. These results also confirm that the appropriate model form was identified across scenarios: the linear model (LM) produced the lowest RMSE in the linear settings, while the nonlinear BART model achieved the best performance in the nonlinear scenario. This pattern provides a useful validation that the proposed spatial whitening step preserves the expected relative performance of linear and nonlinear methods.

As a final note, we state the computation time associated with each model. Overall, the spatial whitening for $n = 50,000$ took 90 seconds on an Apple M1 chip with 64 GB of memory. After the spatial whitening, LM took 4 seconds, RF 49 seconds, BART 88 seconds, boosting 11 seconds, KNN 38 seconds, and SLP 12 seconds. By comparison, the independent method took the same time as the decorrelation approach (no change in input dimensionality) while the spatial basis approach added a negligible amount of computation: LM took 4 seconds, RF 55 seconds, BART 89 seconds, boosting 12 seconds, KNN 43 seconds and SLP 15 seconds. In all, the spatial decorrelation approach takes more time due to the computation associated with the transformation. However, this is a one-time, fixed cost that can be greatly reduced via parallelization.

4 Application

In this section, we demonstrate the performance of our spatial adjustments using an application in pollution monitoring. Importantly, in contrast to the simulations above, this application is not generated from a Gaussian process. Hence, using this application, we wish to see how the spatial adjustment improves ML and DL methods in this setting. Specifically, we analyze data on particulate matter less than 2.5 micrometers ($PM_{2.5}$) in diameter taken from the Environmental Protection Agency's (EPA) network of monitors. After cleaning, the data we consider consists of 593 measurements of $PM_{2.5}$ across the contiguous United States taken on June 5, 2019. We follow Zhan and Datta (2025) and split the data into train and validation sets according to a block-random split strategy which removes whole areas of data and is closer to a real-world scenario (see Appendix S5.1 of Zhan and Datta 2025 for details). This split method resulted in 72 different train-validation splits with a split ratio of approximately 80%-20%. The raw data along with one train-validation split is shown in Figure 1.

Figure 2 displays the median root mean square error (RMSE) across all the train-validation splits using block-random splitting. Notably, for all models, the inclusion of the spatial adjustment decreases RMSE over the independent approach and, with the exception of BART, decreases the RMSE of the basis function approach. Interestingly, there is little difference between the models if spatial correlation is accounted for suggesting that spatial correlation can

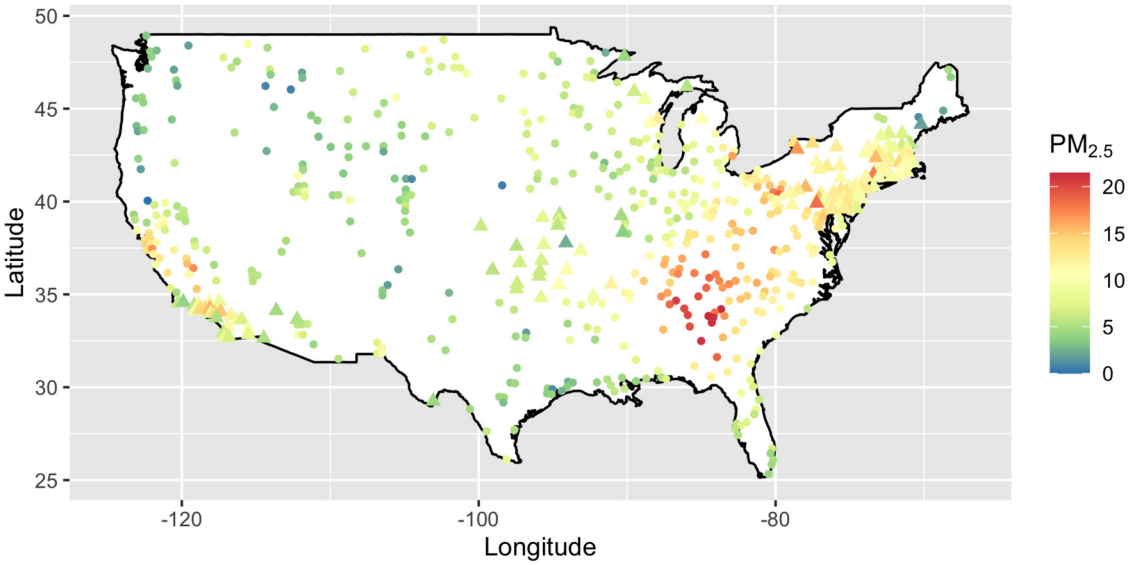

Figure 1: Raw $PM_{2.5}$ data along with an example train-validation split for assessing predictive performance. The validation set is given by the triangle points.

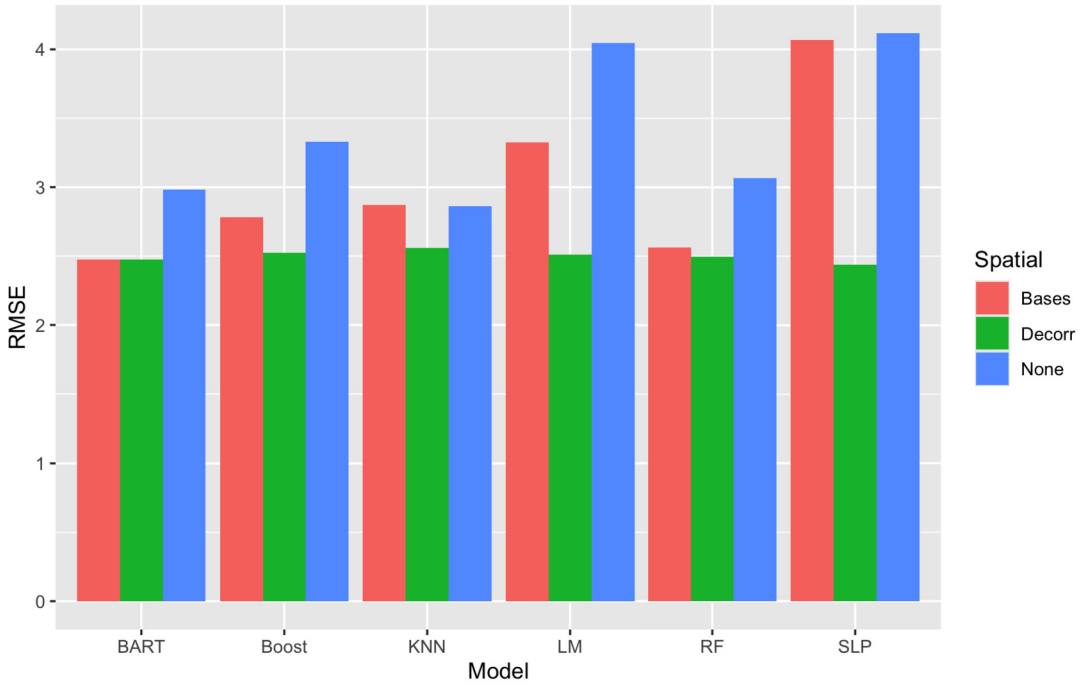

Figure 2: Median RMSE for all models using no spatial adjustment, spatial basis function and the spatial adjustment for the $PM_{2.5}$ example. Note that the spatial adjustment, in all models, decreases the RMSE.

adapt to the type of model being fit to improve predictive ability.

To understand further the impact of accounting for spatial correlation, Figure 3 compares the predictions from the BART model both using and not using the spatial adjustment (we show

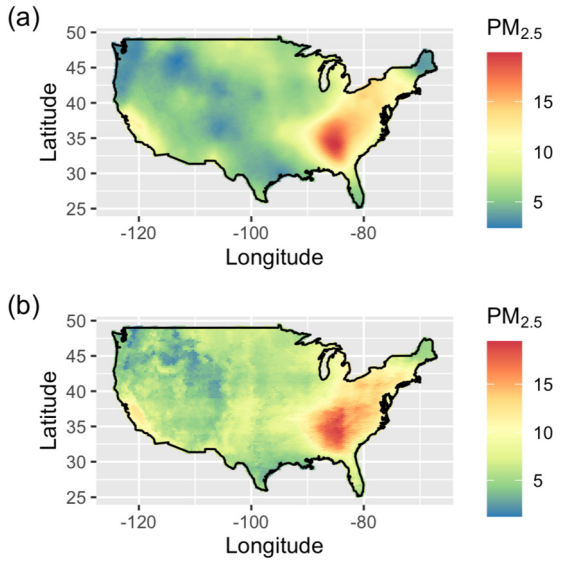

Figure 3: Comparison of predictions from the BART model (a) using and (b) not using the spatial adjustment. The spatially adjusted BART model exhibits smoother predictions than the non-spatial model, reflecting its ability to capture spatial dependence more effectively and yielding a lower RMSE on held-out data.

results from BART here for illustration but all other models had similar results). Clearly, using spatial adjustment leads to smoother spatial predictions. Even without the spatial adjustment, BART can capture some of the spatial structure. However, independent BART had a higher RMSE suggesting that BART may overfit the training data.

5 Conclusions

We present herein a spatial adjustment for using machine and deep learning methods on spatial data. Intuitively, the method works by performing a spatial whitening transformation on the data prior to model fitting, after which predictions are backtransformed to recover the appropriate spatial dependence. The approach can be tuned to match the amount of spatial correlation in the data and is computationally scalable, making it suitable for large spatial datasets. Using both simulated and real data applications, we demonstrated that accounting for spatial correlation improves predictive performance in common machine learning algorithms. While all examples here assume a stationary spatial correlation structure—the most common in practice—the whitening transformation in (2.5) can in principle accommodate any correlation function $\rho(\cdot)$. Extending the approach to more complex structures such as anisotropic or nonstationary correlations remains an open area of research. These extensions would require estimation or tuning of additional parameters beyond the spatial range and nugget considered here.

Notably, the decorrelating transform (2.5) is only valid for quantitative data (note that the theoretical foundation was laid assuming a Gaussian assumption). Obviously, count, Bernoulli, multinomial, or other data types can also exhibit spatial correlation. In future research, we plan to address the issue of non-quantitative spatial data.

Given the breadth of ML and DL approaches, we are not able to do an exhaustive demon-

stration of the spatial adjustment in all approaches. Particularly with DL methods, we only demonstrated how the spatial adjustment works in conjunction with multi-layer perceptrons. As open questions, we need to consider how such a spatial adjustment may work with graphical or convolutional neural networks.

Finally, we note that we did not explore using our approach in conjunction with uncertainty quantification (UQ) for machine and deep learning. This was primarily because UQ methods for machine and deep learning are still under active development with no single agreed-upon method. Under our methods, the approach of, say, Zhang et al. (2020) for random forests or any of the UQ methods for deep learning surveyed in Gawlikowski et al. (2023) could be used to obtain $\widetilde{Y}^{\star}_{\text{lwr}}(\boldsymbol{u})$, a lower bound, and $\widetilde{Y}^{\star}_{\text{upr}}(\boldsymbol{u})$, an upper bound, on the decorrelated data which can then be inverse-transformed via (2.8) to $Y^{\star}_{\text{lwr}}(\boldsymbol{u})$ and $Y^{\star}_{\text{upr}}(\boldsymbol{u})$. While this is a straightforward approach, what remains to be investigated is if such an approach maintains appropriate coverage under the spatial transformation presented here.

Supplementary Material

This material is based upon work supported by the National Aeronautics and Space Administration under Grant/Contract/Agreement No. 10053957-01 and by the National Science Foundation under Grant No. 2053188.

R and Python implementations of the proposed spatial whitening transformation are available as a zip file or at https://github.com/amillane/spatialtransform. The contents are organized as follows:

- **README.md**: A brief overview of the repository structure and usage instructions.
- **R Function/**
  - `Functions/TransformFunctions.R`: R implementation of the whitening and inverse-whitening transformations.
  - `demo.R`: Example code demonstrating use of the R transformation functions.
  - `SimulatedData1.RData`: Example simulated dataset for demonstration.
  - `SimulatedData2.RData`: Second example simulated dataset.
- **Python Function/**
  - `Functions/SpatialTransform.py`: Python implementation of the whitening and inverse-whitening transformations.
  - `Functions/matern.py`: Matern covariance utility functions.
  - `Functions/mknnIndx.py`: Nearest-neighbor index construction for Vecchia approximation.
  - `demo.ipynb`: Jupyter notebook illustrating how to use the Python implementation.
  - `NonLinSimDataSet17.json`: Example nonlinear simulated dataset used in demonstrations.

Together, these materials provide complete code and example data needed to reproduce the spatial whitening transformation and the analyses described in the manuscript.

References

Abdulah S, Ltaief H, Sun Y, Genton MG, Keyes DE (2018). Exageostat: A high performance unified software for geostatistics on manycore systems. *IEEE Transactions on Parallel and Distributed Systems*, 29(12): 2771–2784. https://doi.org/10.1109/TPDS.2018.2850749

Arbia G, Espa G, Giuliani D (2021). *Spatial Microeconometrics*. Routledge.

Banerjee S, Carlin BP, Gelfand AE (2014). *Hierarchical Modeling and Analysis for Spatial Data*. CRC press.

Berrett C, Calder CA (2016). Bayesian spatial binary classification. *Spatial Statistics*, 16: 72–102. https://doi.org/10.1016/j.spasta.2016.01.004

Bishop CM (2006). *Pattern Recognition and Machine Learning*. Springer, New York, NY, USA, 1st edition.

Bradley JR, Cressie N, Shi T (2011). Selection of rank and basis functions in the spatial random effects model. In: *Proceedings of the 2011 Joint Statistical Meetings*, 3393–3406. American Statistical Association, Alexandria, VA.

Bradley JR, Cressie N, Shi T (2016). A comparison of spatial predictors when datasets could be very large. *Statistics Surveys*, 10: 100–131. https://doi.org/10.1214/16-SS115

Chen W, Li Y, Reich BJ, Sun Y (2024). Deepkriging: Spatially dependent deep neural networks for spatial prediction. *Statistica Sinica*, 34: 291–311. https://doi.org/10.5705/ss.202021.0277

Cisneros D, Richards J, Dahal A, Lombardo L, Huser R (2024). Deep graphical regression for jointly moderate and extreme Australian wildfires. *Spatial Statistics*, 100811. https://doi.org/10.1016/j.spasta.2024.100811

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. https://doi.org/10.1080/01621459.2015.1044091

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. https://doi.org/10.1002/wics.1383

Finley AO, Datta A, Cook BD, Morton DC, Andersen HE, Banerjee S (2019). Efficient algorithms for Bayesian nearest neighbor Gaussian processes. *Journal of Computational and Graphical Statistics*, 28(2): 401–414. https://doi.org/10.1080/10618600.2018.1537924

Gawlikowski J, Tassi CRN, Ali M, Lee J, Humt M, Feng J, et al. (2023). A survey of uncertainty in deep neural networks. *Artificial Intelligence Review*, 56(Suppl 1): 1513–1589. https://doi.org/10.1007/s10462-023-10562-9

Gelfand AE, Schliep EM (2016). Spatial statistics and Gaussian processes: A beautiful marriage. *Spatial Statistics*, 18: 86–104. https://doi.org/10.1016/j.spasta.2016.03.006

Georganos S, Grippa T, Niang Gadiaga A, Linard C, Lennert M, Vanhuysse S, et al. (2021). Geographical random forests: A spatial extension of the random forest algorithm to address spatial heterogeneity in remote sensing and population modelling. *Geocarto International*, 36(2): 121–136. https://doi.org/10.1080/10106049.2019.1595177

Gray SD, Heaton MJ, Bolintineanu DS, Olson A (2022). On the use of deep neural networks for large-scale spatial prediction. *Journal of Data Science*, 20(4): 493–511. https://doi.org/10.6339/22-JDS1070

Guinness J (2018). Permutation and grouping methods for sharpening Gaussian process approximations. *Technometrics*, 60(4): 415–429. https://doi.org/10.1080/00401706.2018.1437476

Harris R, Jarvis C (2014). *Statistics for Geography and Environmental Science*. Routledge.

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: 398–425. https://doi.org/10.1007/s13253-018-00348-w

Huang H, Abdulah S, Sun Y, Ltaief H, Keyes DE, Genton MG (2021). Competition on spatial

statistics for large datasets. *Journal of Agricultural, Biological, and Environmental Statistics*, 26: 580–595. https://doi.org/10.1007/s13253-021-00457-z

Katzfuss M, Guinness J (2021). A general framework for Vecchia approximations of Gaussian processes. *Statistical Science*, 36(1). https://doi.org/10.1214/19-STS755

Lin DC, Huang HC, Tzeng S (2023). Some enhancements to deepkriging. *Stat*, 12(1): e559. https://doi.org/10.1002/sta4.559

Lindsay BG, Yi GY, Sun J (2011). Issues and strategies in the selection of composite likelihoods. *Statistica Sinica*, 21(1): 71–105.

Nikparvar B, Thill JC (2021). Machine learning of spatial data. *ISPRS International Journal of Geo-Information*, 10(9): 600. https://doi.org/10.3390/ijgi10090600

Pace RK, Barry R, Sirmans CF (1998). Spatial statistics and real estate. *The Journal of Real Estate Finance and Economics*, 17: 5–13. https://doi.org/10.1023/A:1007783811760

Patelli L, Cameletti M, Golini N, Ignaccolo R (2024). A path in regression random forest looking for spatial dependence: A taxonomy and a systematic review. In: *Advanced Statistical Methods in Process Monitoring, Finance, and Environmental Science: Essays in Honour of Wolfgang Schmid*, Knoth S, Okhrin Y, Otto P, 467–489.

Plant RE (2018). *Spatial Data Analysis in Ecology and Agriculture Using R*. CRC Press.

Saha A, Basu S, Datta A (2023). Random forests for spatially dependent data. *Journal of the American Statistical Association*, 118(541): 665–683. https://doi.org/10.1080/01621459.2021.1950003

Sainsbury-Dale M, Zammit-Mangion A, Richards J, Huser R (2025). Neural Bayes estimators for irregular spatial data using graph neural networks. *Journal of Computational and Graphical Statistics*, 1–16. https://doi.org/10.1080/10618600.2024.2433671

Sauer A, Gramacy RB, Higdon D (2023). Active learning for deep Gaussian process surrogates. *Technometrics*, 65(1): 4–18. https://doi.org/10.1080/00401706.2021.2008505

Sekulić A, Kilibarda M, Heuvelink GB, Nikolić M, Bajat B (2020). Random forest spatial interpolation. *Remote Sensing*, 12(10): 1687. https://doi.org/10.3390/rs12101687

Shaddick G, Zidek JV (2015). *Spatio-Temporal Methods in Environmental Epidemiology*. CRC Press.

Stein A, Gelfand A (2022). The impact of spatial statistics. *Spatial Statistics*, 50: 100641.

Stein ML (2014). Limitations on low rank approximations for covariance matrices of spatial data. *Spatial Statistics*, 8: 1–19. https://doi.org/10.1016/j.spasta.2013.06.003

Tonks A, Harris T, Li B, Brown W, Smith R (2024). Forecasting West Nile virus with graph neural networks: Harnessing spatial dependence in irregularly sampled geospatial data. *GeoHealth*, 8(7): e2023GH000784. https://doi.org/10.1029/2023GH000784

Turner R, Eriksson D, McCourt M, Kiili J, Laaksonen E, Xu Z, et al. (2021). Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In: *NeurIPS 2020 Competition and Demonstration Track*, 3–26. PMLR.

Vu Q, Zammit-Mangion A, Cressie N (2022). Modeling nonstationary and asymmetric multivariate spatial covariances via deformations. *Statistica Sinica*, 32(4): 2071–2093.

Waller LA, Gotway CA (2004). *Applied Spatial Statistics for Public Health Data*. John Wiley & Sons.

Wikle CK, Zammit-Mangion A (2023). Statistical deep learning for spatial and spatiotemporal data. *Annual Review of Statistics and Its Application*, 10: 247–270. https://doi.org/10.1146/annurev-statistics-033021-112628

Wu J, Chen XY, Zhang H, Xiong LD, Lei H, Deng SH (2019). Hyperparameter optimization for machine learning models based on Bayesian optimization. *Journal of Electronic Science and Technology*, 17(1): 26–40.

Yuan Q, Shen H, Li T, Li Z, Li S, Jiang Y, et al. (2020). Deep learning in environmental remote sensing: Achievements and challenges. *Remote Sensing of Environment*, 241, 111716. https://doi.org/10.1016/j.rse.2020.111716

Zammit-Mangion A, Kaminski M.D., Tran B.H., Filippone M, Cressie N (2024). Spatial Bayesian neural networks. *Spatial Statistics*, 60, 100825. https://doi.org/10.1016/j.spasta.2024.100825

Zammit-Mangion A, Ng TLJ, Vu Q, Filippone M (2022). Deep compositional spatial models. *Journal of the American Statistical Association*, 117(540): 1787–1808. https://doi.org/10.1080/01621459.2021.1887741

Zhan W, Datta A (2025). Neural networks for geospatial data. *Journal of the American Statistical Association*, 120(549): 535–547. https://doi.org/10.1080/01621459.2024.2356293

Zhang H, Zimmerman J, Nettleton D, Nordman DJ (2020). Random forest prediction intervals. *The American Statistician*, 74(4): 392–406. https://doi.org/10.1080/00031305.2019.1585288

Ziakopoulos A, Yannis G (2020). A review of spatial approaches in road safety. *Accident Analysis & Prevention*, 135: 105323.