geospaNN: A Python package for geospatial neural networks

Wentao Zhan[1] **and Abhirup Datta**[2]

1 Department of Statistics, University of Wisconsin-Madison **2** Department of Biostatistics, Johns Hopkins Bloomberg School of Public Health

DOI: 10.21105/joss.08389

Software

- Review
- Repository
- Archive

Editor: Vissarion Fisikopoulos

Reviewers:

- @Exelegcho1
- @ArkajyotiSaha

Submitted: 02 May 2025
Published: 29 January 2026

Summary

Geostatistical models are essential for analyzing data with spatial structure across the geosciences, such as climate, ecology, and environmental science. At the same time, modern machine learning methods, especially neural networks (NNs), offer powerful tools for capturing complex, nonlinear relationships. Our package `geospaNN` bridges these two worlds by providing a Python library that integrates NN modeling with scalable spatial statistics. The software enables users to fit flexible spatial regression models, estimate complex mean structures, and generate Gaussian process (GP)–based spatial predictions with uncertainty quantification. Built on the `PyG` library designed for efficient graph neural network (GNN) training, `geospaNN` supports efficient computation on large, irregular spatial datasets. To handle modern geospatial data sizes, `geospaNN` incorporates the Nearest Neighbor Gaussian Process (NNGP) approximation (Datta et al., 2016) for fast covariance computations.

Statement of Need

Researchers in geoscience and related fields frequently need to model relationships among spatially distributed variables and generate reliable spatial predictions. Although many Python machine learning libraries can fit complex nonlinear regression models, they typically ignore spatial correlation, leading to biased estimates and misleading inference when applied to geospatial data. Existing spatial modeling tools in Python provide only partial solutions: some rely on complex neural architectures that sacrifice scientific interpretability, while others use full GP models whose computational demands make them impractical for large datasets.

`geospaNN` addresses these limitations by providing a spatial regression framework that combines the flexibility of NNs with the interpretability and statistical rigor of geostatistical models. It is designed for geoscientists, environmental researchers, and machine learning practitioners who need scalable and principled spatial modeling tools in Python. `geospaNN` enables geometry-aware covariance estimation and spatial prediction at scales—tens of thousands of locations—that are feasible on a personal laptop. This makes advanced spatial analysis accessible to individual researchers without specialized computing infrastructure.

The NNGP implementation within `geospaNN` also fills a notable gap in the Python ecosystem. While widely used R packages such as `spNNGP` (Finley et al., 2019) and `BRISC` (Saha & Datta, 2018) provide efficient NNGP-based spatial models, no comparable Python implementation currently exists. `geospaNN` therefore offers the first Python-based pathway for NNGP modeling in geospatial applications, meeting the growing demand for large-scale spatial analysis.

State of the field

Integrating geospatial data with modern deep learning has motivated the development of several specialized Python tools. For example, `TorchGeo` (Stewart et al., 2022) extends `PyTorch`

(Paszke et al., 2019) for tasks such as land cover classification, object detection, and geospatial segmentation, while the R package `geodl` (Maxwell et al., 2024) was recently introduced for analyzing geospatial and spatiotemporal datasets. However, these frameworks are primarily designed for raster and vector data—especially satellite imagery—rather than for general geostatistical modeling or spatial regression. Their scope is therefore limited when working with point-referenced geospatial data or when statistical interpretability is essential.

For irregular spatial data, GNNs have emerged as a powerful modeling approach. `PyTorch-Geometric` (`PyG`) (Fey & Lenssen, 2019) provides a flexible and efficient framework for implementing GNNs, and these models have been successfully applied to a range of geospatial tasks, including crop yield prediction (Fan et al., 2022) and traffic flow modeling (Wang et al., 2020). Despite their popularity, there is still no unified, statistically oriented GNN software designed specifically for geospatial regression or rigorous covariance modeling. This leaves a gap between machine learning–focused GNN libraries and the needs of statistical geoscience.

GP–based tools provide another major category of spatial modeling software. `PyKrige` (Murphy, 2014) offers classical kriging prediction but is limited to predefined mean functions and lacks scalable covariance computation for large datasets. `GPyTorch` (Gardner et al., 2018) supports flexible mean modeling and GP inference within a mixed-model framework, but its functionality is highly modular and requires substantial custom implementation, making it difficult for general users to apply. Moreover, its covariance approximations are not explicitly designed to exploit spatial geometry, which can reduce efficiency and accuracy compared with approaches tailored to geostatistical structure.

The geospaNN Package

This section provides an overview of the `geospaNN` package, including the model architecture and several technical details. For practical examples and detailed documentation, visit the `geospaNN` website.

NN-GLS Overview

In methodology, `geospaNN` uses NN-GLS (Zhan & Datta, 2025), a novel and scalable class of NNs explicitly designed to account for spatial correlation in the data. NN-GLS embeds NN with the following spatial mixed model:

$$Y(s) = m(X(s)) + \epsilon(s)$$

where $Y(s)$ and $X(s)$ are respectively the outcome and covariates observed at location s, m is a non-linear function relating $X(s)$ to $Y(s)$ to be estimated using a NN. The key distinction from the standard non-linear regression setup is that here the errors $\epsilon(s)$ is a GP that models spatial correlation.

To solve the model, NN-GLS replaces the original loss function with a GLS-style version, which naturally equates to a specialized GNN. For computational efficiency, NNGP is introduced to approximate the covariance,. In NN-GLS, we assume that the parameters of the covariance matrix is unknown. These covariance parameters θ for spatial process $\epsilon(s)$ and the weights parameters of the NN used to model m are estimated iteratively, and training proceeds until the validation loss converges. Once estimation is complete, nearest-neighbor-based kriging is used to generate spatial predictions at new locations.

Core features of `geospaNN`

The `geospaNN` workflow begins by preparing the data and constructing the model inputs. Users may supply covariates, responses, and coordinates in simple matrix form. The function `geospaNN.make_graph` then creates a `DataLoader` object that organizes the data and handles batching efficiently:

```
data = geospaNN.make_graph(X, Y, coord, nn)
```

geospaNN provides flexible tools for specifying neural network architectures and defining training routines, all fully compatible with the PyTorch ecosystem. The code example below illustrates a typical training setup. Here, a two-layer multilayer perceptron is used to model the nonlinear mean structure. The nngls_model object implements the NN–GLS model, and trainer_nngls manages the iterative training process. Users may rely on default hyperparameters or customize them as needed.

```
mlp_nngls = torch.nn.Sequential(
    torch.nn.Linear(p, 100),
    torch.nn.ReLU(),
    torch.nn.Linear(100, 20),
    torch.nn.ReLU(),
    torch.nn.Linear(20, 1),
)
nngls_model = geospaNN.nngls(p=p, neighbor_size=nn, coord_dimensions=2,
                             mlp=mlp_nngls, theta=torch.tensor(theta0))
trainer_nngls = geospaNN.nngls_train(nngls_model, lr=0.1, min_delta=0.001)
training_log = trainer_nngls.train(data_train, data_val, epoch_num= 200,
                                   Update_init=10, Update_step=2,
                                   batch_size = 60, seed = 2025)
```

Once training is complete, the fitted model provides three key capabilities:

1. estimate the non-linear mean function by $\hat{m}$.
2. estimate the spatial parameters by $\hat{\theta}$.
3. predict the outcome at new locations by $\hat{Y}$.

The mean function $m(x)$ represents the non-spatial component of the spatial mixed model, representing the non-spatial relationship between Y and covariates X. To obtain predictions of the mean function for a given matrix of covariates X, users may call:

```
estimate = nngls_model.estimate(X)
```

The estimated spatial parameters $\hat{\theta}$ characterize the spatial correlation structure implied by the model. They are stored internally and can be accessed directly:

```
nngls_model.theta
```

These parameters can be used to reconstruct the implied covariance matrix or inform further geostatistical analyses.

While mean function estimation reflect the connection between variables, to predict the value of response Y at new locations with uncertainty quantification, geospaNN uses predict() method:

```
[test_predict, test_PI_U, test_PI_L] = nngls_model.predict(data_train, data_test,
                                                           PI = True)
```

Other Features

In addition to estimation and prediction for the NN-GLS spatial mixed models, geospaNN offers a suite of additional features that support a wide range of geospatial analyses. geospaNN provides simulation module allowing users to customize the spatial parameters and mean functions to generate Y, X, and s. Users are allowed to customize the spatial coordinates to simulate under different context. geospaNN implements nearest neighbor kriging, an alternate to full kriging, which has been shown in Zhan & Datta (2025) to guarantee accurate prediction interval under various settings. For essential machine learning tasks, geospaNN offers modules including NN architecture design, training log report, and result visualization. geospaNN also implements spatial linear mixed model (SPLMM) as a special case of NN-GLS. It should be an optimal choice for the Python users if efficient SPLMM solution is wanted for large geospatial datasets.

Because the above code snippets rely on additional setup, they are not meant to run independently. Users can find complete, reproducible examples and detailed documentation in the project vignette.

Discussion

The `geospaNN` package provides a machine learning toolkit for geostatistical analysis. Built on an efficient implementation of the NN–GLS approach proposed in Zhan & Datta (2025), `geospaNN` supports a range of core statistical tasks, including nonlinear mean-function estimation, covariance parameter estimation, and spatial prediction with uncertainty quantification. Leveraging the sparsity of the NNGP approximation, the software integrates naturally into the GNN framework, enabling the use of graph-based operations and opening the door to more advanced neural architectures in geospatial modeling.

Despite these strengths, the current version of `geospaNN` has several limitations. At present, the package supports only a limited set of stationary, parametric covariance models and does not handle non-stationary or non-Gaussian spatial processes. The neural network component is designed and tested mainly for simple architectures such as multilayer perceptrons, which work well for moderate-scale spatial data but limit applicability to more complex or high-dimensional input structures. One important future direction is to increase the flexibility of our model and add features to the main steps in `geospaNN` to adopt more general estimation and prediction tasks. In addition, `geospaNN` currently requires R-dependency and does not support GPU acceleration. In the future releases, we will address these key issues to further improve the performance of the software.

Conceptually, a longer-term direction for `geospaNN` is to evolve into a general framework for geospatially informed deep learning, where spatially structured message passing can be incorporated while maintaining statistical interpretability. We also plan to extend the methodology to additional data types and distributional settings beyond the current Gaussian framework.

Acknowledgements

This work is supported by National Institute of Environmental Health Sciences grant R01ES033739. The authors report there are no competing interests to declare.

References

Datta, A., Banerjee, S., Finley, A. O., & Gelfand, A. E. (2016). 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

Fan, J., Bai, J., Li, Z., Ortiz-Bobea, A., & Gomes, C. P. (2022). A GNN-RNN approach for harnessing geospatial and temporal information: Application to crop yield prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, *36*, 11873–11881. https://doi.org/10.1609/aaai.v36i11.21444

Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. *arXiv Preprint arXiv:1903.02428*.

Finley, A. O., Datta, A., Cook, B. D., Morton, D. C., Andersen, H. E., & 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

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., & Wilson, A. G. (2018). GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. *Advances in Neural Information Processing Systems*, *31*.

Maxwell, A. E., Farhadpour, S., Das, S., & Yang, Y. (2024). Geodl: An R package for geospatial deep learning semantic segmentation using torch and terra. *PloS One*, *19*(12), e0315127. https://doi.org/10.31223/x53m6t

Murphy, B. S. (2014). PyKrige: Development of a kriging toolkit for Python. *AGU Fall Meeting Abstracts*, *2014*, H51K–0753.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., & others. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, *32*.

Saha, A., & Datta, A. (2018). BRISC: Bootstrap for rapid inference on spatial covariances. *Stat*, *7*(1), e184. https://doi.org/10.1002/sta4.184

Stewart, A. J., Robinson, C., Corley, I. A., Ortiz, A., Lavista Ferres, J. M., & Banerjee, A. (2022). TorchGeo: Deep learning with geospatial data. *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*, 1–12. https://doi.org/10.1145/3557915.3560953

Wang, X., Ma, Y., Wang, Y., Jin, W., Wang, X., Tang, J., Jia, C., & Yu, J. (2020). Traffic flow prediction via spatial temporal graph neural network. *Proceedings of the Web Conference 2020*, 1082–1092. https://doi.org/10.1145/3366423.3380186

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