University of Brasília
Institute of Exact Sciences
Department of Statistics

Master's Dissertation

Positional Encoder Graph Quantile Neural Networks for Geographic Data

by

William Edward Rappel de Amorim

Brasília, September 2024

Positional Encoder Graph Quantile Neural Networks for Geographic Data

by

William Edward Rappel de Amorim

Dissertation submitted to the Department of Statistics at the University of Brasília in fulfilment of the requirements for obtaining the Master Degree in Statistics.

Advisor: Prof. Dr. Guilherme Souza Rodrigues

Brasília, September 2024

To my family and friends, whose support has been fundamental throughout this journey.

Acknowledgments

Thanks to the faculty of PPGEST/UnB, my advisor Dr. Guilherme Souza Rodrigues, all professors and colleagues who contributed to this work in any way. Also thanks to Prof. Raul Yukihiro Matsushita and Prof. Fabrício Aguiar Silva for accepting the invitation to be part of this work's evaluation committee.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Resumo Expandido

REDES NEURAIS QUANTÍLICAS DE GRAFOS COM CODIFICADOR POSICIONAL PARA DADOS GEOGRÁFICOS

Este trabalho propõe uma nova arquitetura chamada Redes Neurais Quantílicas de Grafos com Codificador Posicional para Dados Geográficos (**PE-GQNN**). Ela foi criada para melhorar a previsão espacial, integrando Redes Neurais de Grafos (GNNs) com Codificação Posicional (PE) e regressão quantílica. A **PE-GQNN** é projetada para superar as limitações dos modelos tradicionais, ao incorporar o contexto espacial e capturar uma ampla gama de quantis condicionais, permitindo a modelagem de incertezas de maneira mais precisa.

Os dados geoespaciais, ou dados espaciais, contêm informações que incluem um ou mais atributos relacionados às coordenadas geográficas dos dados, geralmente definidas por latitude e longitude. Estes dados são amplamente utilizados em várias áreas, como economia, meteorologia, transporte urbano, redes sociais, plataformas de comércio eletrônico, entre outras. Um aspecto importante dos dados espaciais é a autocorrelação espacial, que se refere ao fenômeno em que pontos de dados espaciais não são estatisticamente independentes, mas sim correlacionados, especialmente em locais próximos.

Modelos de regressão espacial tradicionais utilizam matrizes de pesos espaciais ou termos de defasagem espacial para capturar a estrutura espacial nos dados. No entanto, esses mode-

los frequentemente não conseguem modelar relações complexas entre as variáveis preditoras e a variável de interesse. Por outro lado, os Processos Gaussianos (GPs), que assumem uma distribuição preditiva Gaussiana, apresentam alta flexibilidade e podem capturar relações complexas, mas enfrentam desafios significativos de escalabilidade e complexidade computacional, especialmente para conjuntos de dados grandes.

As GNNs surgiram recentemente como uma solução poderosa e escalável para aplicar redes neurais a dados estruturados em grafos. A abordagem tradicional de GNNs para dados espaciais consiste em representar os dados como um grafo, permitindo que as GNNs sejam aplicadas para aprendizado de representações e inferência. No entanto, a eficácia desta abordagem depende fortemente de como os dados espaciais são representados no grafo, e a aplicação tradicional de GNNs a dados espaciais pode não modelar efetivamente relações espaciais complexas.

Para melhorar o desempenho preditivo, foi proposta a Rede Neural de Grafos com Codificador Posicional (PE-GNN), uma abordagem modular e flexível projetada para aplicação a dados espaciais. A PE-GNN constrói o grafo usando distâncias calculadas a partir das coordenadas geográficas e incorpora um Codificador Posicional (PE) para assimilar o contexto espacial de cada par de coordenadas. A saída do PE resulta em um *embedding* espacial que é concatenado com as variáveis explicativas dos nós antes da aplicação do operador GNN, que produz uma previsão pontual para a variável alvo.

Entretanto, a PE-GNN está limitada a fornecer apenas previsões pontuais e não foi projetada para oferecer uma descrição probabilística completa da distribuição condicional da variável alvo. Para superar essa limitação, este trabalho propõe a **PE-GQNN**, que combina a abordagem PE-GNN com inovações que permitem a quantificação de incerteza ao realizar previsões, oferecendo uma descrição completa da distribuição preditiva condicional.

A **PE-GQNN** inclui três inovações principais: (1) uma nova arquitetura que funde o procedimento em dois passos proposto por Kuleshov and Deshpande (2022) em um único modelo intrinsecamente calibrado, adiando a concatenação do quantil τ proposto por Si, Kuleshov, and Bishop (2022); (2) uma alteração estrutural que aplica o operador GNN apenas às caracterísi-

cas dos nós, que são então concatenadas com o *embedding* espacial; e (3) a introdução da média da variável alvo dos vizinhos do nó no grafo como uma característica adicional em uma camada próxima ao output.

A **PE-GQNN** foi testada em três conjuntos de dados reais contendo dados espaciais. Para o conjunto de dados da Califórnia, a **PE-GQNN** atingiu desempenho de estado da arte, com melhorias relevantes em relação a GNN tradicional e PE-GNN. Ao compará-la com o modelo Processo Neural Condicional com Multi-Atenção Espacial (SMACNP), proposto por Bao, Zhang, and Zhang (2024), a **PE-GQNN** consistentemente apresentou desempenho preditivo superior com uma grande vantagem: melhor quantificação de incerteza.

Utilizando esse mesmo conjunto de dados, as contribuições de cada inovação são reveladas ao realizar uma análise aprofundada dos resultados. A inovação τ, que corresponde à aplicação do framework de regressão quantílica proposto por Si, Kuleshov, and Bishop (2022), realizou um efeito de regularização e melhorou notavelmente a calibração das previsões de quantis, reduzindo o MPE e o MADECP. A inovação estrutural, que envolve a aplicação do operador GNN apenas nas características, é fundamental para melhorar o desempenho das previsões e melhorar a calibração, conforme evidenciado pela redução do MSE, MAE, MPE e MADECP. Além disso, o uso da média dos alvos dos vizinhos de treinamento como uma característica introduzida em uma das últimas camadas da rede, inspirada no método KNN, também melhorou ainda mais o modelo em termos de desempenho preditivo e calibração, apresentando uma redução substancial de MSE, MAE, MPE e MADECP. Além disso, foi verificado empiricamente que as predições da **PE-GQNN** não apresentaram casos de cruzamento de quantis.

Os resultados experimentais para os outros dois conjuntos de dados demonstraram que a **PE-GQNN** consistentemente supera a GNN tradicional e a PE-GNN em todas as arquiteturas de GNN. Em cada conjunto de dados, as inovações da **PE-GQNN** levam a reduções significativas em MSE, MAE e MPE. Por outro lado, para o conjunto de dados Air Temperature, o SMACNP alcança o menor MSE e MAE, mas sofre com previsões significativamente descalibradas, refletidas por um MPE e MADECP muito mais altos em comparação com a **PE-GQNN**.

Em resumo, a **PE-GQNN** representa um avanço significativo na modelagem de dados espaciais, combinando a flexibilidade das GNNs com a capacidade de quantificar incerteza através da regressão quantílica, oferecendo maior capacidade preditiva e uma descrição probabilística completa da distribuição condicional da variável de interesse.

Palavras-chave: Redes Neurais de Grafos. Regressão quantílica. Dados geoespaciais. Quantificação de incerteza. Calibração. Recalibração de Modelos.

Abstract

Positional Encoder Graph Neural Networks (PE-GNNs) are a leading approach for modeling continuous spatial data. However, they often fail to produce calibrated predictive distributions, limiting their effectiveness for uncertainty quantification. We introduce the Positional Encoder Graph Quantile Neural Network (**PE-GQNN**), a novel method that integrates PE-GNNs, Quantile Neural Networks, and recalibration techniques in a fully nonparametric framework, requiring minimal assumptions about the predictive distributions. We propose a new network architecture that, when combined with a quantile-based loss function, yields accurate and reliable probabilistic models without increasing computational complexity. Our approach provides a flexible, robust framework for conditional density estimation, applicable beyond spatial data contexts. We further introduce a structured method for incorporating a KNN predictor into the model while avoiding data leakage through the GNN layer operation. Experiments on benchmark datasets demonstrate that **PE-GQNN** significantly outperforms existing state-of-the-art methods in both predictive accuracy and uncertainty quantification.

Keywords: Graph Neural Networks (GNNs). Quantile regression. Geospatial data. Uncertainty quantification. Calibration. Model recalibration.

Contents

List of Tables

List of Figures

Abbreviations and Acronyms

CNN	Convolutional Neural Network
CNP	Conditional Neural Process
ECP	Empirical Cumulative Probability
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GNN	Graph Neural Network
GP	Gaussian Process
GSAGE	Graph Sample and Aggregate
MAE	Mean Absolute Error
MADECP	Mean Absolute Distance of the Empirical Cumulative Probability
MLE	Maximum Likelihood Estimate
MPE	Mean Pinball Error
MSE	Mean Squared Error
NLP	Natural Language Processing
NN	Neural Network
OLS	Ordinary Least Squares
PE	Positional Encoder
PE-GNN	Positional Encoder Graph Neural Network
PE-GQNN	Positional Encoder Graph Quantile Neural Network
SMACNP	Spatial Multi-Attention Conditional Neural Process

List of Symbols and Notations

τ	probability associated with the prediction at quantile regression, lies in $(0, 1)$
n	sample size
n_B	batch size
p	number of features
i	observation index, goes from 1 to n or 1 to n_B
k	number of nearest neighbors for the k-nearest neighborhood
K	number of GNN layers
$\boldsymbol{A}$	the graph adjacency matrix
$\boldsymbol{D}$	the graph degree matrix
y_i	target variable value for the i-th observation
$\hat{y}_i$	target variable prediction value for the i-th observation
$\boldsymbol{x}_i$	vector of features for the i-th observation
$\boldsymbol{c}_i$	vector of geographic coordinates for the i-th observation
$\mathbb{1}(.)$	indicator function, returns 1 if the event is true and 0 if false
$\hat{q}_i(\tau)$	target variable prediction value for the i-th observation τ-probability quantile

Chapter 1

Introduction

Large spatial datasets are collected in a wide range of applications in economics (Anselin, 2022), meteorology (Bi et al., 2023), urban transportation (Lv et al., 2014; Derrow-Pinion et al., 2021; Kashyap et al., 2022), social networks (Xu et al., 2020), e-commerce (Sreenivasa and Nirmala, 2019) and other fields. Gaussian Processes (GPs) (Rasmussen and Williams, 2006; Cressie and Wikle, 2011) are a fundamental tool for modelling spatial data on continuous domains. They are flexible and interpretable models for unknown functions, both in spatial and more general regression settings. However, with time complexity $O(n^3)$ and storage complexity $O(n^2)$, naive GP methods quickly become intractable for large datasets. This has led to a large range of approximate inference methods, such as those based on sparse approximations to covariance or precision matrices (Reinhard Furrer and Nychka, 2006; Lindgren, Rue, and Lindström, 2011), low rank approximations (Cressie, Sainsbury-Dale, and Zammit-Mangion, 2022) or nearest neighbour approximations (Vecchia, 1998; Datta et al., 2016; Katzfuss and Guinness, 2021).

Given the difficulty of GP computations, it's of interest to explore scalable methods for large spatial datasets using neural networks (NNs) and to enhance their ability to quantify uncertainty. A state-of-the-art method for making spatial predictions using Graph Neural Networks (GNNs) is the Positional Encoder Graph Neural Network (PE-GNN) of Klemmer, Safir, and Neill, 2023.

Our contribution is to make three key modifications to the PE-GNN architecture to enhance its ability to make accurate spatial predictions and to quantify uncertainty. These modifications will be explained further below.

NNs are popular in data modeling and prediction tasks like computer vision and natural language processing (NLP). However, traditional NNs struggle to handle spatial dynamics or graph-based data effectively. GNNs (Kipf and Welling, 2017; Veličković et al., 2018; Hamilton, Ying, and Leskovec, 2017) offer a powerful and scalable method for applying NNs to graph-structured data. The idea is to share information through the edges of a graph, allowing nodes to exchange information during learning. GNNs are versatile and can uncover nonlinear relationships among inputs, hidden layers, and each node's neighborhood information. The success of GNNs in spatial applications largely depends on the spatial graph construction, including choice of distance metric and the number of neighboring nodes, and traditional GNNs often struggle to model complex spatial relationships. To address this, Klemmer, Safir, and Neill, 2023 introduced the PE-GNN, which enhances predictive performance in spatial interpolation and regression. However, PE-GNN is not designed to provide a full probabilistic description of the target's distribution, and assuming a Gaussian distribution for predictions can lead to poorly calibrated intervals, such as 80% intervals that fail to contain the true outcome 80% of the time. Recently, Bao, Zhang, and Zhang (2024) proposed a new framework called Spatial Multi-Attention Conditional Neural Processes (SMACNPs) for spatial small sample prediction tasks. SMACNPs use GPs parameterized by NNs to predict the target variable distribution, which enables precise predictions while quantifying the uncertainty of these predictions.

Methods based on quantile regression are an alternative approach to probabilistic forecasting making rapid progress in recent years. Si, Kuleshov, and Bishop (2022) introduced a novel architecture for estimating generic quantiles of a conditional distribution, proposing a set of objective functions that lead to enhancements in density estimation tasks. In one dimension, this method produces quantile function regression and cumulative distribution function regression. Kuleshov and Deshpande (2022) argue that the method of Si, Kuleshov, and Bishop (2022) is

inefficient with high-dimensional predictors. To address this, they modify the original formulation to incorporate a post hoc recalibration procedure whereby an auxiliary model recalibrates the predictions of a trained model. The first model outputs features, usually summary statistics like quantiles, representing a low-dimensional view of the conditional distribution. The auxiliary model, the recalibrator, uses these features as input to produce calibrated predictions using Si *et al.*'s quantile function regression framework. The main drawback is that it requires training two separate models, each needing its own training set.

Our work makes three contributions. (1) We propose a new architecture that merges the two-step procedure of Kuleshov and Deshpande (2022) into a single model by postponing the concatenation of the τ value proposed by Si, Kuleshov, and Bishop (2022). In this way, we enhance the network's ability to model uncertainty and introduce a regularization mechanism. The model becomes robust to high-dimensional predictor spaces, even though few assumptions are made about the form of the target's conditional distribution. This change allows a single model to fully describe the predictive conditional distribution and to generate quantile predictions and prediction intervals as byproducts. It can be applied to any context, not just spatial regression or GNNs. We show how to integrate this strategy into the PE-GNN framework to create an intrinsically calibrated model with no extra computational cost. (2) We introduce a structural change to PE-GNN. Instead of applying the GNN operator to the concatenation of the nodes' features and the spatial embedding, we apply it only to the features. (3) In PE-GNN, the GNN operator uses neighbours' features to create new node representations but does not include the target value of neighboring nodes. Our third contribution introduces the mean target value of a node's neighbours as a feature after the GNN layers, closer to the output. This allows the model to use neighboring observations of the target variable when making predictions.

The structure of this work is as follows: Section 2 offers a brief background overview, Section 3 outlines the proposed method for geographic data prediction, Section 4 shows experimental results on three real-world datasets, and Section 5 concludes.

Chapter 2

Background

In this section, we provide a comprehensive overview of the theoretical foundations and recent advancements relevant to our proposed approach. This background sets the stage for introducing our novel **PE-GQNN** in the following section.

2.1 Positional Encoder

The Transformer architecture, as introduced by Vaswani et al. (2017), stands as a transformative paradigm shift in the domain of NLP models. This architectural innovation pivots from the conventional sequence-to-sequence models by leveraging the power of self-attention mechanisms. By employing multi-layer stacks of attention modules, it reimagines the way dependencies within sequential data are captured. This pivotal concept of "attention" enables the model to weigh the significance of various elements within a sequence, offering an elegant solution to the long-range dependency problem encountered in recurrent models. With the incorporation of positional encodings, the Transformer effectively addresses the sequential order of data, making it a foundation in modern NLP models and reaching state-of-the-art performances in many NLP benchmarks.

Inspired by the Transformer architecture (Vaswani et al., 2017) for geographic data (Mai

et al., 2020), PE-GNN (Klemmer, Safir, and Neill, 2023) employs a PE composed of two components: a sinusoidal transformation and a fully-connected NN. The first component is a deterministic transformation formed by the concatenation of sinusoidal functions, incorporating variations in frequency and scale. Let $\boldsymbol{C}_B = [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{n_B}]^\top$ be the matrix containing the spatial coordinates of a batch of datapoints, typically of dimension $n_B \times 2$, where each $\boldsymbol{c}_i$ corresponds to the pair $(\text{latitude}_i,\ \text{longitude}_i)$. Then, the sinusoidal transformations are given by

$$\begin{aligned} ST(\boldsymbol{C}_B, \sigma_{min}, \sigma_{max}) &= [ST_0(\boldsymbol{C}_B, \sigma_{min}, \sigma_{max}); \ldots; ST_{S-1}(\boldsymbol{C}_B, \sigma_{min}, \sigma_{max})] \\ \text{where } ST_s(\boldsymbol{C}_B, \sigma_{min}, \sigma_{max}) &= [ST_{s,1}(\boldsymbol{C}_B, \sigma_{min}, \sigma_{max}); ST_{s,2}(\boldsymbol{C}_B, \sigma_{min}, \sigma_{max})] \\ \text{and } ST_{s,v}(\boldsymbol{C}_B, \sigma_{min}, \sigma_{max}) &= \left[\cos\left(\frac{\boldsymbol{C}_B^{[v]}}{\sigma_{min} g^{s/(S-1)}}\right); \sin\left(\frac{\boldsymbol{C}_B^{[v]}}{\sigma_{min} g^{s/(S-1)}}\right)\right] \\ & \forall s \in \{0, \ldots, S-1\}, \forall v \in \{1, 2\}; \end{aligned} \tag{2.1}$$

with σ_{min} and σ_{max} being the minimum and maximum grid scales, S being the number of grid scales considered and $g = \frac{\sigma_{max}}{\sigma_{min}}$. Consequently, the sinusoidal transformation handles the spatial dimensions (typically represented as latitude and longitude) separately. The second component is a fully-connected NN, denoted by $NN(\Theta_{PE})$. This network takes the output produced by the sinusoidal transformation as its input and subsequently processes it through a fully-connected NN. This transformation results in the desired vector space representation, thereby generating the coordinate embedding matrix $\boldsymbol{C}_B^{emb} = PE(\boldsymbol{C}_B, \sigma_{min}, \sigma_{max}, \Theta_{PE}) = NN(ST(\boldsymbol{C}_B, \sigma_{min}, \sigma_{max}), \Theta_{PE})$.

2.2 Graph

A graph is an exceedingly useful representation for describing various phenomena involving connections among its components, such as social networks, biological molecules, urban transportation networks, and more. A graph $G = (V, E)$ is formed by a set of vertices (or nodes) $V = \{v_1, \ldots, v_n\}$ and a set of edges (or links) $E = \{e_1, \ldots, e_m\}$. Nodes represent the

entities within the graph, while edges represent the relationships between the entities (Pósfai and Barabasi, 2016).

There are several ways to classify graphs, with the three main criteria being the number of entity types, the direction of connections, and the weight of connections.

In the first criterion, if all edges represent the same type of entity (e.g., individuals), then the graph is considered homogeneous. Conversely, if different types of entities exist within the same graph (e.g., movies and actors), it is classified as heterogeneous.

Regarding the direction of edges, a graph can be directed, meaning its connections have a defined direction, leading from a source node to a destination node. However, graphs can also be undirected, where no specific direction is associated with the link; it merely represents a relationship between nodes, without the notion of "origin" and "destination".

Lastly, concerning the weight of links, a graph can be categorized as unweighted if all its edges lack a defined property associated with the link, merely indicating their existence or absence. Conversely, weighted graphs assign a weight to each edge to quantify the relevance of that connection. These weights can measure factors such as the distance between nodes or their similarity.

A crucial concept in graph representation is the adjacency matrix, $\boldsymbol{A}$. This matrix is used to represent the connectivity of the graph, indicating which nodes are connected and the weight of each connection. It is a square matrix with dimensions $n \times n$. For unweighted graphs, the matrix entries are binary: 1 if there is a connection and 0 otherwise. In weighted graphs, matrix entries are equal to the weight assigned to that connection, and if there is no connection, they receive a value of 0. Additionally, undirected graphs have symmetric adjacency matrices, as $a_{ij} = a_{ji},\ \forall i, j \in \{1, \ldots, n\}$.

Another important concept in graph representation is the degree matrix, $\boldsymbol{D}$. The degree matrix is a diagonal matrix used to represent the degree of each node in the graph. For a graph with n nodes, the degree matrix has dimensions $n \times n$. The entry d_{ii} on the diagonal of the degree matrix corresponds to the degree of node i. For unweighted graphs, this entry is simply

the number of edges connected to node i. In weighted graphs, it is the sum of the weights of the edges connected to node i. All off-diagonal entries in the degree matrix receive a value of 0.

2.3 Graph Neural Network

GNNs are powerful and scalable solutions for representation learning and inference with graph-structured data. They are capable of leveraging the existing topological structure of correlation between nearby nodes in the graph and represent each node in a latent space embedding suitable for the specific downstream task at hand (Wu et al., 2022). The most popular GNN architectures use this graph structure to update the embeddings of each node, to consider not only the features of each node but also those of its neighbors, in an **Aggregate** and **Combine** iterative process (Wu et al., 2022).

The first step consists of the aggregation of the features from neighbors of each node, which creates a unified output that encapsulates the holistic context of each node's immediate surroundings. This can be done in a wide range of ways, such as feature summation, averaging, or more sophisticated methodologies (such as attention mechanisms).

Subsequent to the aggregation operation, the **Combine** step comes into play, aiming to combine each node's prior representation with the output of the **Aggregate** step. The **Combine** process can be as straightforward as concatenating the representations followed by a linear transformation, or it can employ more intricate techniques such as nonlinear functions.

The initial embedding of each node is its feature vector, so $\boldsymbol{H}_B^{(0)} = \boldsymbol{X}_B$, i.e., $\boldsymbol{h}_v^{(0)} = \boldsymbol{x}_v$, $\forall v \in V_B$. Then, for each GNN layer $k \in \{1, \ldots, K\}$, an iteration of the **Aggregate** and **Combine** process is executed, defined as

$$\boldsymbol{a}_v^{(k)} = AGG^{(k)}(\{\boldsymbol{h}_u^{(k-1)} : u \in \mathcal{N}(v)\}), \tag{2.2}$$

$$\boldsymbol{h}_v^{(k)} = COMB^{(k)}(\{\boldsymbol{h}_v^{(k-1)}, \boldsymbol{a}_v^{(k)}\}),\ \forall v \in V_B, \tag{2.3}$$

where $\mathcal{N}(v)$ is the set of neighbors of node v. The most popular GNN architectures follow the **Aggregate** and **Combine** process, but they differ in the way they aggregate neighbors messages and update the embeddings.

Graph Convolutional Networks (GCNs) (Kipf and Welling, 2017) are inspired in the convolution operation from Convolutional Neural Networks (CNNs), which are very famous for their use in image processing and computer vision, which yielded state-of-the-art performances in this research domain (Krizhevsky, Sutskever, and Hinton, 2012). For weighted graphs, GCN layers have the following update equations

$$\boldsymbol{H}_B^{(k)} = f^{(k)}\left(\boldsymbol{D}_B^{-1/2}\left[\boldsymbol{A}_B + \boldsymbol{I}_B\right]\boldsymbol{D}_B^{-1/2}\boldsymbol{H}_B^{(k-1)}\boldsymbol{W}^{(k)}\right), \text{ for } k \in \{1, \ldots, K\}. \tag{2.4}$$

Here, $f^{(k)}$ is the activation function (e.g., ReLU) and $\boldsymbol{W}^{(k)}$ is a matrix of learnable parameters.

Another popular GNN architecture is the Graph Attention Network (GAT) (Veličković et al., 2018), which leverages the attention mechanism in the GNNs domain to empower nodes to prioritize their most informative neighbors, a departure from GCNs, which treat all neighbors uniformly. GATs calculate pairwise attention scores between nodes, allowing them to discern the importance of each neighbor's information. Subsequently, these scores are employed to weight the contribution of neighbors features during node feature aggregation. Furthermore, they can utilize multiple attention heads to capture diverse relational information among nodes. For weighted graphs, GAT layers with single-headed attention only would have the following update equations

$$\boldsymbol{h}_v^{(k)} = f^{(k)}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \alpha_{vu}^{(k)} \boldsymbol{W}^{(k)} \boldsymbol{h}_u^{(k-1)}\right), \ \forall v \in V_B. \tag{2.5}$$

The attention weights $\alpha^{(k)}$ are generated by an attention mechanism $\boldsymbol{AT}^{(k)}$, normalized so that

the sum over all neighbors of each node v is 1:

$$\alpha_{vu}^{(k)} = \frac{\boldsymbol{AT}^{(k)}\left(\boldsymbol{h}_v^{(k-1)}, \boldsymbol{h}_u^{(k-1)}, a_{v,u}\right)}{\sum_{w \in \mathcal{N}(v) \cup \{v\}} \boldsymbol{AT}^{(k)}\left(\boldsymbol{h}_v^{(k-1)}, \boldsymbol{h}_w^{(k-1)}, a_{v,w}\right)}, \ \forall (v,u) \in E_B. \tag{2.6}$$

Here, $f^{(k)}$ is the activation function (e.g., ReLU), $a_{v,u}$ is the (v,u) entry of $\boldsymbol{D}_B^{-1/2}\left[\boldsymbol{A}_B + \boldsymbol{I}_B\right]\boldsymbol{D}_B^{-1/2}$, $\boldsymbol{W}^{(k)}$ and $\boldsymbol{AT}^{(k)}$ are potentially learnable parameters.

Hamilton, Ying, and Leskovec (2017) proposed the GSAGE architecture, which has many similarities with GCN, but is more flexible and scalable. One of the key aspects of GSAGE is the neighborhood sampling: regardless of the actual size of a node's neighborhood, GSAGE takes a fixed-size random sample of the neighborhood, enabling their application on exceptionally large graphs. The update equations of this approach are defined as

$$\boldsymbol{h}_v^{(k)} = f^{(k)}\left(\boldsymbol{W}^{(k)}\left[\underset{u \in \mathcal{N}(v)}{AGG}(\{\boldsymbol{h}_u^{(k-1)}\}), \boldsymbol{h}_v^{(k-1)}\right]\right), \ \forall v \in V_B. \tag{2.7}$$

Here, $f^{(k)}$ is the activation function (e.g., ReLU), $\boldsymbol{W}^{(k)}$ and AGG are potentially learnable parameters. If AGG is chosen as the sum, then the update equations become very similar to the original GCN formulation by Kipf and Welling (2017).

2.4 Positional Encoder Graph Neural Network

Klemmer, Safir, and Neill (2023) proposed a novel approach for applying GNNs to spatial data: PE-GNN. It is a powerful and flexible approach for predictive modeling of geographic data. It is based on GNNs and highly modular, capable of accommodating any GNN backbone, including all the GNN layers presented in Section 2.3. Through the implementation of several innovations in the traditional method of using GNNs with spatial data, the empirical studies presented by Klemmer, Safir, and Neill (2023) show that PE-GNN consistently improved the performance of different GNN layers and achieved competitive results compared to models

based on GPs.

To achieve these objectives, the first proposed innovation was to incorporate a PE into the spatial coordinates processing. In the traditional approach of GNNs with geographic data, the use of coordinates is limited to computing distances between datapoints, which are used to construct the graph with a defined number of nearest neighbors. Once the graph is constructed, the traditional approach no longer uses spatial location at any point. In contrast, in addition to using coordinates to construct the graph, PE-GNN also uses them to learn an embedding containing the spatial context of each pair of coordinates. To achieve this objective, the coordinate pair passes through the PE, as described in Section 2.1. In this process, the PE takes the set of spatial coordinates for each datapoint as input and produces a vector representing the learned spatial embedding. This vector is then column concatenated with the node features before the application of the GNN operator. Thus, for a given batch B of randomly sampled datapoints, the input to the first GNN layer is

$$\boldsymbol{H}_B^{(0)} = concat\left(\boldsymbol{X}_B, \boldsymbol{C}_B^{emb}\right). \tag{2.8}$$

Another innovation used in PE-GNN is to learn an auxiliary task in parallel with the main task. In the traditional approach, the GNN operator takes as input the graph constructed from the coordinates and the features of each node and outputs the prediction of each node's target variable. In contrast, PE-GNN receives as input the graph, along with the features of each node and the embedding constructed by the PE, and, in addition to the main task prediction, it also predicts the Local Moran's I, as proposed by Klemmer and Neill (2021). Local Moran's I is a measure of spatial autocorrelation for the target variable y_i, given by

$$I_i = (n-1)\frac{(y_i - \bar{y})}{\sum_{j=1, j\neq i}^{n} (y_j - \bar{y})^2} \sum_{j=1, j\neq i}^{n} a_{i,j}(y_j - \bar{y}), \tag{2.9}$$

where $\bar{y}$ is the sample mean of y, and $a_{i,j} \in \boldsymbol{A}_B$ denotes adjacency of observations i and j.

The third innovation lies in the training process, as PE-GNN uses a batch-based procedure. At each training step, a random batch B of nodes is sampled, given by $p_1, \ldots, p_{n_B} \in B$. Using only the nodes belonging to the batch, the entire process of constructing the training graph, generating the spatial embedding, column concatenating with the features, and applying the GNN operator is carried out. Furthermore, the target Local Moran's I is constructed using only the data in the batch, which, according to Klemmer, Safir, and Neill (2023), allows for the minimization of the problem known as the Moran's I scale sensitivity. An additional advantage of this training strategy is that it learns a more generalized spatial embedding since, at each training step, the same datapoint can have a completely different neighborhood. The PE is jointly learned with the other parameters of PE-GNN, in contrast to the unsupervised approach proposed by Mai et al. (2020). The loss function used by Klemmer, Safir, and Neill (2023) is based on MSE and given by

$$\mathcal{L}_B = MSE\left(\hat{\boldsymbol{y}}_B, \boldsymbol{y}_B\right) + \lambda MSE\left(I(\hat{\boldsymbol{y}}_B), I(\boldsymbol{y}_B)\right), \tag{2.10}$$

where λ denotes the auxiliary task weight.

Despite being a powerful and flexible approach for representation learning and inference of geographic data, PE-GNN has a major drawback: it is configured for generating point predictions rather than supplying a comprehensive probabilistic representation of the conditional distribution. Although one could presume a Gaussian distribution with the predicted values at its center, this assumption is often not met, resulting in imprecise probabilistic and quantile forecasts.

2.5 Calibration and recalibration

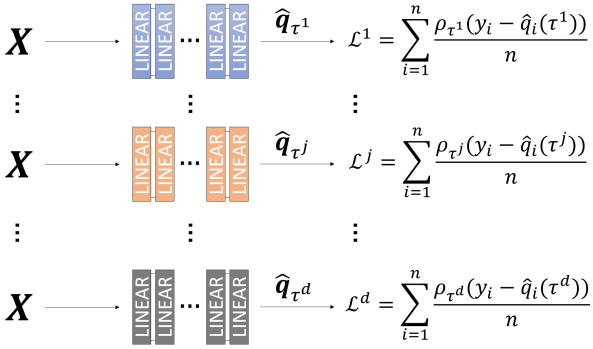
Quantifying uncertainty is crucial when using predictive models, especially in the case of NNs, where it presents a complex challenge. Calibration is a desired property of predictive

models because it allows us to trust their probabilistic predictions. A calibrated model makes predictions where the predicted confidence levels match the actual outcomes. For example, an 80% confidence level prediction should contain the true observed value 80% of the time. Essentially, a calibrated model estimates the probabilities of events such that the observed frequencies of these events match the predicted probabilities.

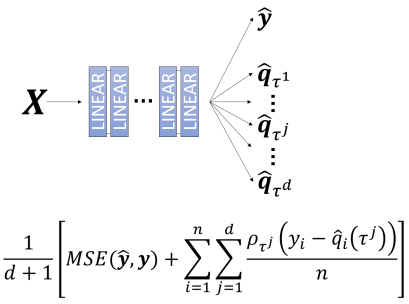
Traditional machine learning models often do not produce calibrated probabilistic predictions (Niculescu-Mizil and Caruana, 2005; Guo et al., 2017). To improve this, a recalibration process can be applied to modify the original predictions, making them calibrated. Platt scaling (Platt et al., 1999) is a recalibration method that achieves calibrated predictions but is limited to quantile calibration. Similarly, Niculescu-Mizil and Caruana (2005) introduced isotonic regression for recalibration, which is also effective but restricted to quantile calibration. Guo et al. (2017) introduced temperature scaling, an adaptation of Platt scaling (Platt et al., 1999). This method adjusts the network's logits by dividing them by a positive scalar $T > 0$, followed by applying the softmax function. However, Kumar, Liang, and Ma (2019) argued that widely used recalibration techniques such as Platt scaling and temperature scaling are not as well-calibrated as commonly believed. They proposed the scaling-binning calibrator, which combines binning with parametric function fitting to minimize variance and improve calibration accuracy. Song et al. (2019) proposed a more advanced approach using complex variational approximations for distribution calibration, which, while innovative, is limited to the recalibration of Gaussian distributions.

2.6 Quantile regression

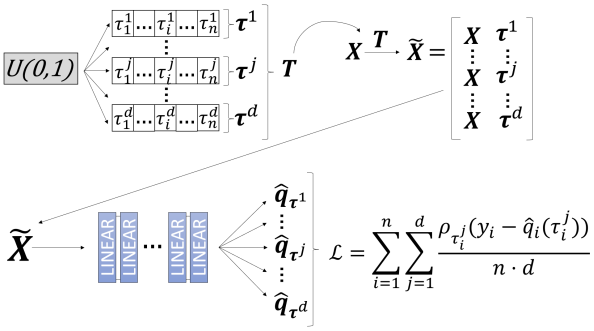
Traditional Gaussian linear regression, known as OLS regression, seeks to estimate parameters that, when used in a linear combination with the explanatory variables, produce predictions that minimize the MSE on the training data. Behind the scenes, this method assumes that the distribution of the target variable, when conditioned on the explanatory variables, follows



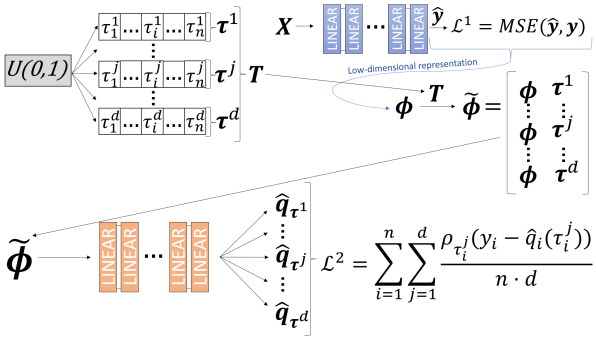
(a) Non-linear quantile regression using NN.



(b) Non-linear multiple quantile regression.



(c) Non-linear quantile function regression.



(d) Two-step density estimation.

Figure 2.1: (a) For each quantile of interest, a separate NN is trained. (b) Rodrigues and Pereira (2020): one NN outputs $d + 1$ predictions: one for the expectation and d for the quantiles. (c) Si, Kuleshov, and Bishop (2022): a single NN trained to predict *any* generic quantile of the conditional distribution. (d) Kuleshov and Deshpande (2022): two-step procedure: the first model outputs a low-dimensional representation of the conditional distribution, which a recalibrator then uses to produce calibrated predictions.

a Gaussian distribution with the mean given by the linear combination of features and their respective coefficients, and the variance being an unknown constant to be estimated. In summary, OLS regression states that, for each observation $i \in \{1, \ldots, n\}$, we have $Y_i|\boldsymbol{X}_i \sim Gaussian(\boldsymbol{X}_i\boldsymbol{\beta}, \sigma^2)$. We obtain the parameters estimates $\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^{p+1}} \sum_{i=1}^{n}(y_i - \boldsymbol{X}_i\boldsymbol{\beta})^2/n$ and point predictions $\hat{y}_i = \hat{\mathbb{E}}[Y_i|\boldsymbol{X}_i] = \boldsymbol{X}_i\hat{\boldsymbol{\beta}}$. As the Gaussian distribution assumption is very often violated, this approach may not be suitable in certain situations.

Often, the study's interest is not in estimating the mean of the response but rather some quantile of the conditional distribution, such as the median or another quantile. For these situations, or when the assumptions of OLS regression are violated, Koenker and Bassett Jr (1978) proposed linear quantile regression, which is an extension of OLS regression. It still uses the same linear combination for prediction but replaces the MSE with the pinball loss function, given by

$$\rho_\tau(r_i) = \begin{cases} \tau r_i & \text{if } r_i \geq 0, \\ (\tau - 1)r_i & \text{if } r_i < 0, \end{cases} \tag{2.11}$$

where $r_i = y_i - \hat{q}_i(\tau)$ and $\hat{q}_i(\tau) = \boldsymbol{X}_i\hat{\boldsymbol{\beta}}$. Here, τ is the desired cumulative probability associated with the quantile and $\hat{q}_i(\tau)$ is the predicted quantile. Equation 2.11 provides the pinball loss (also known as tilted loss) for the i-th observation. To compute the loss for the whole dataset, it is necessary to compute the average of each data point's loss. Then, coefficient estimates are obtained by

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^{p+1}}{\arg\min} \sum_{i=1}^{n} \rho_\tau \left(y_i - \boldsymbol{X}_i\boldsymbol{\beta}\right)/n. \tag{2.12}$$

Despite being more flexible than OLS regression, linear quantile regression requires a linear relationship between the features and the chosen quantile of the target variable, which is a very restrictive assumption.

With the rise of NNs for various prediction tasks, a natural extension of quantile linear regression has emerged: non-linear quantile regression. This concept is illustrated in Figure

2.1a, where we seek to estimate the quantile associated with τ^1. The target to be predicted and the loss function remain identical to those in the linear case (Equation 2.11). However, instead of using a linear combination of features, we use the NN framework. By adding one or more layers with non-linear transformations, NNs provide greater flexibility to the model and can improve predictive performance.

With this approach, we eliminate the need for the two previous assumptions (Gaussian distribution and linearity). However, this method has a major drawback: for each quantile we aim to estimate, it is necessary to train an independent model with weights learned separately from the models of other quantiles. This is illustrated in Figure 2.1a, where if we are interested in the prediction of d quantiles, we must train d models. Consequently, each model specializes in estimating a single quantile without sharing representation learning, leading to inconsistent quantile predictions that often present the quantile crossing problem (e.g., a median prediction below the prediction for the first quartile).

In the previous two quantile regression approaches, a single quantile of interest must be defined, and thus, the trained model specializes in estimating only the chosen quantile. However, in many situations, there is an interest in simultaneously estimating various quantiles or even the entire conditional distribution with only one single model. Rodrigues and Pereira (2020) proposed an approach that modifies the architecture of the NNs used for predicting a single quantile to output multiple predictions: one for the expectation and one for each quantile of interest. For example, if the goal is to predict the mean and d different quantiles, the network would have $d + 1$ output values, as shown in Figure 2.1b. Then, the loss function is defined as follows:

$$\mathcal{L} = \frac{1}{d+1}\left[\text{MSE}\left(\hat{\boldsymbol{y}}, \boldsymbol{y}\right) + \sum_{i=1}^{n}\sum_{j=1}^{d}\frac{\rho_{\tau^j}\left(y_i - \hat{q}_i(\tau^j)\right)}{n}\right]. \tag{2.13}$$

This loss function combines the traditional MSE with d values of the pinball loss: instead of evaluating each datapoint only once, each datapoint is used $d + 1$ times, where d is the number of quantiles to be predicted. This strategy allows all the quantiles of interest to be estimated si-

multaneously and provides a discrete approximation of the cumulative distribution of the target variable. However, to estimate quantiles not directly predicted by the network, it is necessary to interpolate the estimated quantiles or perform some additional treatment. Moreover, this strategy is not capable of fully estimating the conditional density because it is limited to those previously defined quantiles of interest. As shown by Rodrigues and Pereira (2020), it still exhibits, to some extent, the quantile crossing problem.

In order to create an even more flexible non-linear quantile regression model based on neural networks, Si, Kuleshov, and Bishop (2022) proposed a novel method that aims to generate a model that is independent of an arbitrary selection of quantiles. This procedure is presented in Figure 2.1c. For each datapoint sampled during training, d values of τ, where $\tau \sim U(0, 1)$, are sampled. Each of these d values is concatenated with the datapoint features to obtain a quantile estimate. Therefore, for each datapoint, there are d predicted quantiles, one for each of the τ values sampled from the Uniform distribution. The loss function is apparently similar to the one described in Equation 2.13, but they predict random quantiles and discard the MSE component:

$$\mathcal{L} = \frac{1}{n \cdot d} \sum_{i=1}^{n} \sum_{j=1}^{d} \rho_{\tau_i^j} \left(y_i - \hat{q}_i(\tau_i^j) \right). \tag{2.14}$$

Here, they do not consider a single or a predefined set of quantiles, but quantiles sampled randomly. Consequently, as the network learns, it becomes capable of providing a direct estimate to *any* quantile of interest. Therefore, this procedure outputs an inherently calibrated model suitable for conditional density estimation. However, Kuleshov and Deshpande (2022) argue that this method proposed by Si, Kuleshov, and Bishop (2022) is not efficient in moderate to high dimensions of the predictor space.

Kuleshov and Deshpande (2022) used the architecture proposed by Si, Kuleshov, and Bishop (2022) to create a two-step procedure. This approach retains the advantages of the original method introduced by Si, Kuleshov, and Bishop (2022) while making it robust and suitable for larger predictor spaces. The approach is presented in Figure 2.1d.

The first step is to train a model that receives the features as its inputs and outputs a set of features, usually summary statistics (e.g., quantiles), which constitute a low-dimensional representation of the conditional distribution. Then, using a separate dataset called the calibration set, an auxiliary model (usually referred to as a recalibrator) is trained. The recalibrator receives this set of features as input, together with d sampled values of τ, where $\tau \sim U(0, 1)$, and outputs calibrated predictions.

In the procedure proposed by Si, Kuleshov, and Bishop (2022), a single model takes as input the original features and τ values. For situations with a large number of predictors, this results in an inefficient model. Using the modification proposed by Kuleshov and Deshpande (2022), the τ values are introduced only during the training of the recalibrator, which also takes as input a low-dimensional vector representing the conditional distribution predicted by the first model. This strategy allows the recalibrator to understand how this representation can be used to calibrate the predictions of the first model.

The main disadvantages of this approach are: (1) it requires training two NNs and (2) it is highly dependent on the choice of features that constitute the low-dimensional representation of the conditional distribution.

2.7 Spatial Multi-Attention Conditional Neural Processes

Proposed by Bao, Zhang, and Zhang (2024), SMACNPs are devised to tackle the complexities of spatial prediction tasks, especially when there are limited observed samples and numerous prediction samples. Traditional methods like GPs effectively measure interpolation uncertainty but face significant computational challenges with larger sample sizes. Standard NNs, while scalable, often overfit when dealing with small sample sizes. SMACNPs combine the advantages of GPs and NNs to enhance spatial small sample prediction tasks.

SMACNPs use a modular design that applies various attention mechanisms to extract pertinent information from different types of sample data. Task representation is derived by evaluat-

ing the spatial correlation between sample points and the relationships within attribute variables. GPs parameterized by NNs predict the distribution of the target variable, providing precise predictions and quantifying uncertainty. This approach integrates spatial context and correlations into the model, delivering state-of-the-art results in spatial small sample prediction tasks regarding both predictive performance and reliability.

Solutions like Kriging methods use GPs-based models for spatial prediction, but these require careful parameter selection and can underperform with sparse observed points. Spatial heterogeneity, where the distribution is non-stationary or anisotropic, also demands models adaptable to local environments. SMACNPs address these issues by employing attention mechanisms to separately model spatial dependencies and attribute influences.

The SMACNP framework comprises a mean encoder and a variance encoder. The mean encoder features two modules: one modeling spatial autocorrelation using Laplace attention and another capturing relationships between attribute variables and target values with multi-head attention. The variance encoder assesses uncertainty based only on explanatory attributes and spatial locations, improving robust estimations by removing any target value influence. The decoder then aggregates contextual information to predict the target distribution.

In summary, SMACNPs combine the strengths of GPs and NNs to deliver accurate spatial predictions, making them an effective tool for spatial small sample prediction tasks by providing both high predictive accuracy and uncertainty estimates. However, SMACNPs may fail when the conditional distributions deviate from the Gaussian assumption. Despite this limitation, experiments on simulated and real-world datasets demonstrate that SMACNPs surpass existing methods in spatial small sample prediction tasks.

Chapter 3

Method

In this work, we propose a novel approach to spatial data prediction tasks: the Positional Encoder Graph Quantile Neural Network (**PE-GQNN**). Algorithm 1 shows the step-by-step procedure to train a **PE-GQNN** model, and Figure 3.1 illustrates its complete pipeline. Here, each rectangle labeled "GNN" and "LINEAR" represents a set of one or more neural network layers, with the type of each layer defined by the title inside the rectangle. At each layer, a nonlinear transformation (e.g. ReLU) may be applied. Each datapoint p_i comprises three components $p_i = \{y_i, \boldsymbol{x}_i, \boldsymbol{c}_i\}$. The component y_i is the target variable, and as the focus here is regression, then y_i is a continuous scalar. Additionally, $\boldsymbol{x}_i$ is the feature vector and $\boldsymbol{c}_i$ contains the geographical coordinates associated with observation i.

After initializing the model and hyperparameters, the first step of **PE-GQNN** is to randomly sample a batch B of datapoints, $p_1, \ldots, p_{n_B}$. The batch can be fully represented by the target $\boldsymbol{y}_{B(n_B \times 1)}$, features $\boldsymbol{X}_{B(n_B \times p)}$, and coordinates matrices $\boldsymbol{C}_{B(n_B \times 2)}$, respectively. The next step uses the matrix of geographical coordinates $\boldsymbol{C}_B = [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{n_B}]^\top$ to obtain spatial embeddings for each datapoint (Algorithm 1, Step 5). This process receives $\boldsymbol{C}_B$ as input, and after passing through deterministic sinusoidal transformations and a fully-connected NN, outputs the spatial embedding matrix of the batch ${\boldsymbol{C}_B^{emb}}_{(n_B \times u)}$, containing the spatial context of each pair of coordinates. $\boldsymbol{C}_B$ is also used to compute the distance between each pair of datapoints (Algorithm

Algorithm 1 PE-GQNN training

Require:

Training data target, features, and coordinates matrices: $\boldsymbol{y}_{(n\times 1)}$, $\boldsymbol{X}_{(n\times p)}$, and $\boldsymbol{C}_{(n\times 2)}$.
A positive integer k defining the number of neighbors considered in the spatial graph.
Positive integers $tsteps$ and n_B, the number of training steps and the batch size.
Positive integers u, g, and s, the embedding dimensions considered in, respectively, the PE, the GNN layers, and the layer where we introduce $\boldsymbol{\tau}$ and $\bar{\boldsymbol{y}}$.
An activation function $f(\cdot)$ for $\boldsymbol{\tau}$.

Ensure:

A set of learned weights for the model initialized at Step 1.

1: Initialize model with random weights and hyperparameters.
2: Set optimizer with hyperparameters.
3: **for** $b \leftarrow 1$ **to** $tsteps$ **do** ▷ Batched training
4: Sample minibatch B of n_B datapoints: $\boldsymbol{X}_{B(n_B\times p)}$, $\boldsymbol{C}_{B(n_B\times 2)}$, $\boldsymbol{y}_{B(n_B\times 1)}$.
5: Input $\boldsymbol{C}_{B(n_B\times 2)}$ into PE, which outputs the batch's spatial embedding matrix $\boldsymbol{C}^{emb}_{B\ (n_B\times u)}$.
6: Compute the great-circle distance between each pair of datapoints from $\boldsymbol{C}_B$.
7: Construct a graph using k-nearest neighbors from the distances computed in Step 6.
8: Set $\boldsymbol{A}_B$ as the adjacency matrix of the graph constructed in Step 7.
9: **for** $i \leftarrow 1$ **to** n_B **do**
10: Using $\boldsymbol{A}_B$, compute $\bar{y}_i = \frac{1}{k}\sum_{j=1}^{k} y_j$, where $j = 1, \ldots, k$ are the neighbors of i.
11: **end for**
12: Set $\bar{\boldsymbol{y}}_B = [\bar{y}_1, \ldots, \bar{y}_{n_B}]^\top$.
13: Apply GNN layers to the features $\boldsymbol{X}_{B(n_B\times p)}$, followed by fully-connected layers to reduce dimensionality. This step outputs a feature embedding matrix $\boldsymbol{X}^{emb}_{B\ (n_B\times g)}$.
14: Column concatenate $\boldsymbol{X}^{emb}_{B\ (n_B\times g)}$ with $\boldsymbol{C}^{emb}_{B\ (n_B\times u)}$, which results in $\boldsymbol{L}_{B(n_B\times(g+u))}$.
15: Apply fully-connected layers to reduce $\boldsymbol{L}_{B(n_B\times(g+u))}$ to $\boldsymbol{\phi}_{B(n_B\times s)}$.
16: Create a vector with values sampled from $U(0,1)$: $\boldsymbol{\tau}_{B(n_B\times 1)} = [\tau_1, \ldots, \tau_{n_B}]^\top$.
17: Column concatenate $\boldsymbol{\phi}_B$ with $f(\boldsymbol{\tau}_B)$ and $\bar{\boldsymbol{y}}_B$ to create $\widetilde{\boldsymbol{\phi}}_{B(n_B\times(s+2))}$.
18: Predict the target quantile vector $[\hat{q}_1(\tau_1), \ldots, \hat{q}_{n_B}(\tau_{n_B})]^\top$ using $\widetilde{\boldsymbol{\phi}}_B$.
19: Compute loss $\mathcal{L}_B = \frac{1}{n_B}\sum_{i=1}^{n_B} \rho_{\tau_i}\left(y_i - \hat{q}_i(\tau_i)\right)$.
20: Update the parameters of the model using stochastic gradient descent.
21: **end for**

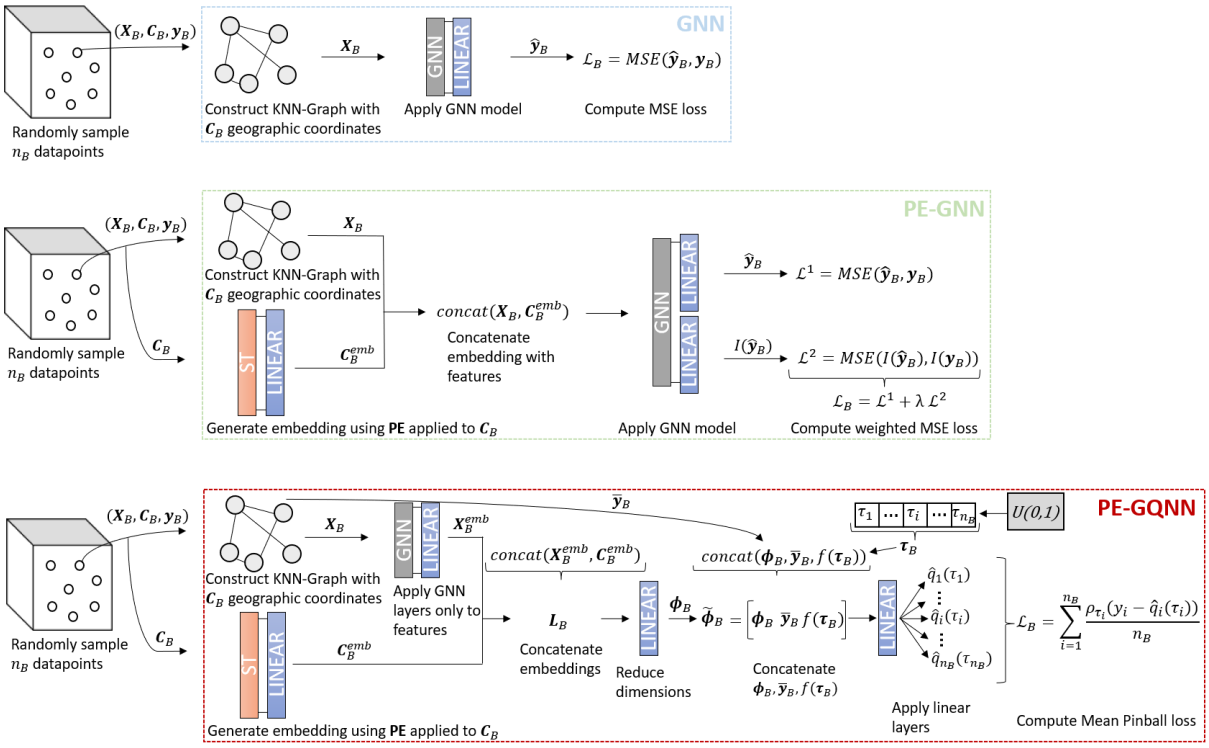


Figure 3.1: PE-GQNN compared to PE-GNN and GNN.

1, Step 6). From these distances and a predefined number of nearest neighbors, a graph can be constructed, with each datapoint as a node and edge weights computed from the distances, leading to the batch adjacency matrix $\boldsymbol{A}_B$.

At Step 13 of Algorithm 1, the first distinction between **PE-GQNN** and PE-GNN arises: instead of using the concatenation of the feature matrix and the spatial embedding as the input for the GNN operator, we apply the GNN operator only to the feature matrix $\boldsymbol{X}_B$. One or more fully-connected layers are then used to reduce the feature embedding dimensionality. This process receives the constructed graph and the batch feature matrix $\boldsymbol{X}_{B(n_B \times p)}$ as input and yields an embedding matrix of features as output: $\boldsymbol{X}_B^{emb}{}_{(n_B \times g)}$. This modification applies the GNN operators exclusively to the features, without smoothing out the PEs. The GNN layers can be of any desired type. Step 14 of Algorithm 1 performs a column concatenation between the feature embedding $\boldsymbol{X}_B^{emb}{}_{(n_B \times g)}$ and the output obtained from the PE: $\boldsymbol{C}_B^{emb}{}_{(n_B \times u)}$ (Figure 3.1). This concatenation results in the matrix $\boldsymbol{L}_{B(n_B \times (g+u))}$. After that, the other innovations of

PE-GQNN come into play.

First, we use one or more fully-connected layers (Algorithm 1, Step 15) to reduce the dimensionality of $\boldsymbol{L}_B$, making it suitable for two of the three innovations in **PE-GQNN**. This set of fully-connected layers outputs the matrix $\boldsymbol{\phi}_{B(n_B \times s)}$, which is then combined with $\bar{\boldsymbol{y}}_B$ and $\boldsymbol{\tau}_B$. $\bar{\boldsymbol{y}}_B$ represents a vector with one scalar for each datapoint in the batch, containing the mean target variable among the training neighbours for each node. It is computed using the graph constructed in previous steps (Algorithm 1, Step 10), and has dimensions $n_B \times 1$. It is comparable to a vector of predictions generated by a KNN regression model, where neighbours are determined using the distance calculated from geographical coordinates. Here, we used the simple average due to its relationship with KNN prediction; however, one could use a weighted average via the adjacency matrix $\boldsymbol{A}_B$. We introduce this input at a later stage to avoid data leakage. If the GNN operator received $\bar{\boldsymbol{y}}_B$ as input, after completing the message passing process in each GNN layer, the true node target value would inadvertently be transmitted to its neighbours, creating potential data leakage (Appleby, Liu, and Liu, 2020).

In the same layer where $\bar{\boldsymbol{y}}_B$ is introduced, we apply a similar approach to Si, Kuleshov, and Bishop (2022) to make **PE-GQNN** an inherently calibrated model suitable for probabilistic and quantile predictions. For each batch B, we create a $n_B \times 1$ vector $\boldsymbol{\tau}_{B(n_B \times 1)} = [\tau_1, \ldots, \tau_{n_B}]^\top$ of random $U(0, 1)$ draws (Algorithm 1, Step 16). Then, we column concatenate $\boldsymbol{\phi}_B$ with $f(\boldsymbol{\tau}_B)$ and $\bar{\boldsymbol{y}}_B$ to create $\widetilde{\boldsymbol{\phi}}_{B(n_B \times (s+2))}$ (Algorithm 1, Step 17), where $f(\cdot)$ is an activation function. Here we propose use of $f(\cdot) = logit(\cdot)$, to facilitate the network's learning. Subsequently, forward propagation is computed (Algorithm 1, Step 18) in one or more fully-connected layers, outputting predicted quantiles for each datapoint in the batch. The batch loss is the one proposed by Si, Kuleshov, and Bishop (2022), but with $d = 1$ for the τ values.

This procedure aims to improve the model's ability to learn the conditional probability distribution of the target variable, enhancing uncertainty estimation and quantile predictions. Instead of introducing τ values alongside features at the network's input, as suggested by Si, Kuleshov, and Bishop (2022), we delay their entry into a reduced latent dimension to boost learning. This

adjustment makes **PE-GQNN** suitable for both low- and high-dimensional predictor spaces. It also improves on the Kuleshov and Deshpande (2022) approach by merging the two-network process into a single, intrinsically calibrated model.

Incorporating τ values into the model architecture improves its ability to model uncertainty and serves as a regularization mechanism (Rodrigues and Pereira, 2020). The use of pinball loss for quantile regression acts as a natural regularizer, producing a detailed description of the predictive density beyond just mean and variance estimation. For predictions, the quantile of interest, τ, must be given, along with the basic data components (e.g. $\tau = 0.25$ gives the first quartile). If interest is in predicting multiple quantiles for the same observation, the input can be propagated up to the layer where $\boldsymbol{\tau}$ is introduced. For each quantile of interest, propagation can be limited to the final layers.

Target domain The final layer should use an activation function coherent with the domain of the target variable, ensuring model outputs are valid for target distribution support. E.g., an exponential function could be appropriate if the target variable is continuous, unbounded and positive.

Quantile crossing This phenomenon occurs when estimated quantile functions for different quantile levels (τ) intersect, violating the requirement that higher quantiles be greater than or equal to lower quantiles. In **PE-GQNN**, by utilizing the same latent representation up to the layer where the quantile level (τ) is introduced, the architecture adopts a hard-parameter sharing multi-task learning framework. This severely mitigates the problem of quantile crossings by constraining the flexibility of independent quantile regression neural network models. If τ is introduced at the prediction layer, it is guaranteed that quantile crossings will be absent, as the layer equation would be

$$\hat{q}_i(\tau) = f\left(bias + w_\tau \tau + w_{\bar{y}_i}\bar{y}_i + \sum_{j=1}^{neurons} w_j u_j\right),\ \forall i \in 1, \ldots, n_B. \tag{3.1}$$

Here, $neurons$ denotes the number of neurons in the prediction layer, excluding τ and $\bar{y}_i$. $bias$, w_τ, $w_{\bar{y}_i}$, and $\{w_j\}$ are the prediction layer parameters, and $\{u_j\}$ are the activation values from the previous layer. Commonly, f is chosen to be monotonic, resulting in a monotonic relationship between τ and $\hat{q}_i(\tau)$. When τ is introduced at a layer proximal to, but preceding, the prediction layer, the results in Section 4 suggest our approach is not prone to suffer from quantile crossing.

Number of Monte Carlo samples When applying the framework proposed by Si, Kuleshov, and Bishop (2022), we chose to use $d = 1$ for the τ values. Let $\mathcal{L}(\theta, \tau, \boldsymbol{x}, y)$ be the loss function for a given quantile $\tau \sim \mathrm{U}(0, 1)$ and an observed pair $(\boldsymbol{x}, y) \sim D_{\text{data}}$, where D_{data} denotes the full data generative process. On each training iteration, we minimize $\mathcal{L}_B$, which, by the Law of Large Numbers, converges to $\tilde{\mathcal{L}}(\theta) = \mathbb{E}_{\tau,\boldsymbol{x},y}\mathcal{L}(\theta, \tau, \boldsymbol{x}, y)$, as the batch size, n_B, goes to infinity. Therefore, the gradients converge to the same value for any d, provided that $n_B \to \infty$. This choice $(d = 1)$ simplifies the implementation without sacrificing performance, as shown in Section 4.

Chapter 4

Experiments

4.1 Experimental setup

PE-GQNN was implemented using PyTorch (Paszke et al., 2019) and PyTorch Geometric (Fey and Lenssen, 2019). The source code is available at: `https://github.com/WilliamRappel98/PE-GQNN`. We conducted comprehensive simulations to explore the prediction performance and other properties of the proposed model, comparing it with state-of-the-art methods. Computation was performed on an Intel i7-7500U processor with 16 GB of RAM, running Windows 10.

Candidate models The experiment was designed to compare six primary approaches for addressing spatial regression problems across three distinct real-world datasets. Table 4.1 lists each candidate model and their applicable datasets. All models were trained using the Adam optimizer (Kingma and Ba, 2015), early stopping and, for all GNN-based models, $k = 5$ nearest-neighbours.

Approach I involves the traditional application of GNNs to geographic data. Three types of GNN layers were considered: GCNs (Kipf and Welling, 2017), GATs (Veličković et al., 2018), and GSAGE (Hamilton, Ying, and Leskovec, 2017). For each of these, the architecture remains consistent to facilitate performance comparisons: two GCN/GAT/GSAGE layers with ReLU

Table 4.1: Summary of candidate models.

Approach	Model	Type	Components					Loss	Datasets
			PE	Moran's I	τ	Structure	$\bar{y}$		
I	GNN	GNN	No	No	No	No	No	MSE	All
II	PE-GNN $\lambda =$ best	GNN	Yes	Yes	No	No	No	$\text{MSE}_y + \lambda\text{MSE}_{I(y)}$	All
III	PE-GQNN τ	GNN	Yes	No	Yes	No	No	Pinball	California
IV	PE-GQNN τ, Structure	GNN	Yes	No	Yes	Yes	No	Pinball	California
V	PE-GQNN	GNN	Yes	No	Yes	Yes	Yes	Pinball	All
VI	SMACNP	GP	No	No	No	No	No	Log Likelihood	All but 3D road

activation and dropout, followed by a linear prediction layer.

Approach II involves the application of PE-GNN (Klemmer, Safir, and Neill, 2023) with optimal weights for each dataset and layer type combination, as demonstrated by the experimental findings of Klemmer, Safir, and Neill (2023). The GNN architecture used is the same as for approach I. It was implemented using the code available at: `https://github.com/konstantinklemmer/pe-gnn`.

Approach III is similar to PE-GNN but augmented with the first innovation proposed in this study: the quantile regression framework described in Section 3 is applied. Approach IV is similar to III, but augmented with an additional innovation: the structural alteration in the model's architecture, where the GNN operator is applied only to the features. Approach V, which is the primary focus of this research, explores the utilization of **PE-GQNN**. The PE and GNN layers' architectures remain identical to the previous approaches, with any alterations limited to the proposed innovations.

Finally, a benchmark approach that does not use GNNs but was recently proposed for modelling spatial data will be considered as approach VI: SMACNPs. This approach, proposed by Bao, Zhang, and Zhang (2024), has demonstrated superior predictive performance, surpassing GPs models in the three real-world datasets considered. This model was implemented following the specifications of Bao, Zhang, and Zhang (2024), using the code available at: `https://github.com/bll744958765/SMACNP`.

Approaches I and II do not inherently provide predicted conditional distributions. However, as they optimize the MSE metric, they implicitly learn a Maximum Likelihood Estimate (MLE)

of a Gaussian model. Thus, the predictive distribution considered for these approaches was a Gaussian distribution centered on the point prediction with variance equal to the MSE of the validation set. For computational simplicity in the experiments, instead of calculating $\bar{\boldsymbol{y}}_B$ for each batch, we pre-calculated $\bar{\boldsymbol{y}}$ using the entire training set.

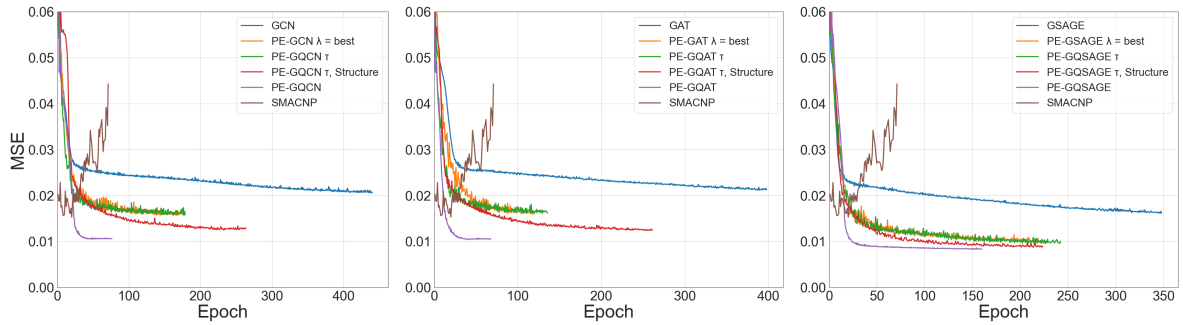
Performance metrics We evaluate predictive accuracy using Mean Squared Error (MSE) and Mean Absolute Error (MAE). To assess calibration of the predictive distributions, we use Mean Pinball Error, MPE $= \frac{1}{n}\sum_{i=1}^{n} \rho_{\tau_i}\left(y_i - \hat{q}_i(\tau_i)\right)$, where $\tau_i \sim U(0,1)$, and the Mean Absolute Distance of the Empirical Cumulative Probability,

$$\text{MADECP} = \frac{1}{99}\sum_{j=1}^{99}\left|\tau^j - \frac{1}{n}\sum_{i=1}^{n}\mathbb{1}\left[y_i \leq \hat{q}_i(\tau^j)\right]\right|. \tag{4.1}$$

For quantile predictions of a calibrated model for a given τ, the proportion of observed values less than or equal to the predicted quantile should approximate τ. Evaluating the MADECP helps determine whether the predicted quantiles are accurate and consistent across the entire space.

4.2 California Housing

This dataset comprises pricing information for >20,000 residential properties in California, recorded during the 1990 U.S. census (Pace and Barry, 1997). The main objective is a regression task: predict housing prices, y, through the incorporation of six predictive features, $\boldsymbol{x}$, and geographical coordinates, $\boldsymbol{c}$. The predictive features are neighborhood income, house age, number of rooms, number of bedrooms, occupancy and population. All models were trained and evaluated using 80% of the data for training, 10% for validation, and 10% for testing. In the case of SMACNP, to adhere to the specifications of Bao, Zhang, and Zhang (2024), a training subsample was extracted to represent 10% of the entire dataset. The validation MSE curves throughout training are shown in Figure 4.1. The number of training epochs and final



(a) GCN. **(b)** GAT. **(c)** GSAGE.

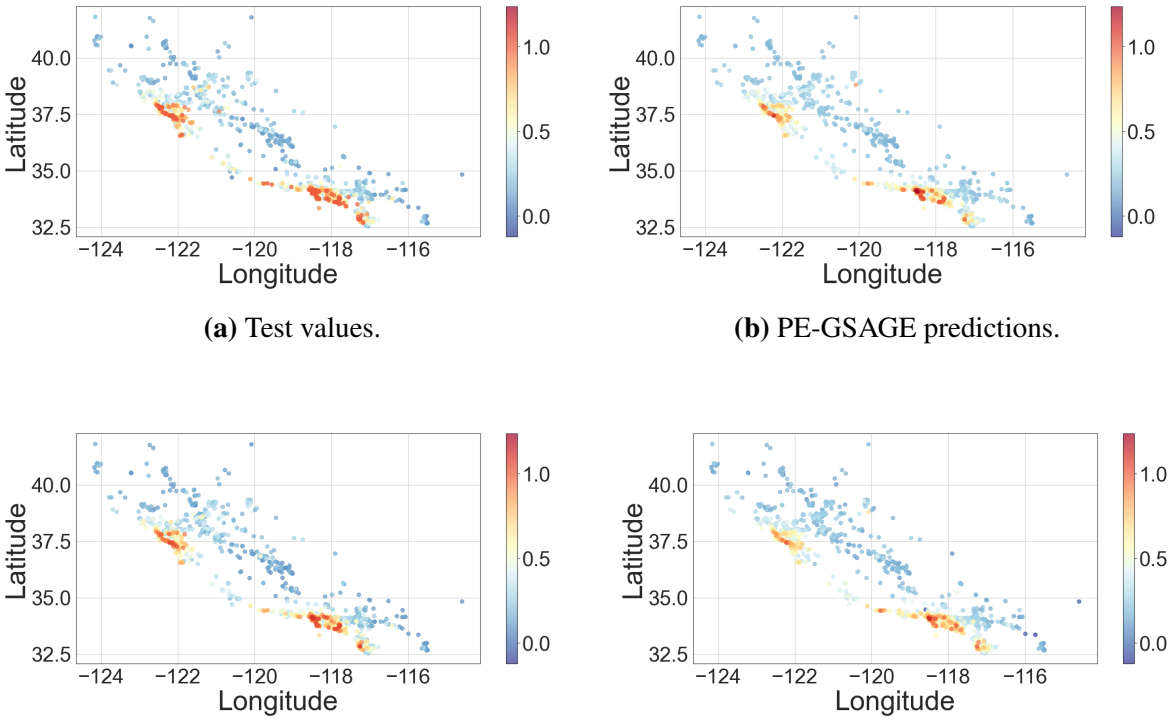
Figure 4.1: Validation error curves on the California Housing dataset, measured by the MSE metric.

test dataset performance metrics are in Table 4.2.

Table 4.2: Performance metrics on the California Housing test set.

Model	**Epochs**	**Parameters**	**MSE**	**MAE**	**MPE**	**MADECP**
GCN	441	**1,313**	0.0222	0.1101	0.0403	0.0475
PE-GCN λ = best	170	24,129	0.0179	0.0935	0.0354	0.0450
PE-GQCN τ	179	25,217	0.0179	0.0914	0.0351	0.0428
PE-GQCN τ, Structure	264	26,169	0.0138	0.0800	0.0302	0.0287
PE-GQCN	76	26,201	0.0114	0.0686	0.0272	0.0262
GAT	398	1,441	0.0227	0.1099	0.0410	0.0586
PE-GAT λ = best	120	24,290	0.0183	0.0930	0.0352	0.0476
PE-GQAT τ	136	25,345	0.0179	0.0926	0.0355	0.0413
PE-GQAT τ, Structure	261	26,297	0.0140	0.0829	0.0312	**0.0193**
PE-GQAT	**68**	26,329	0.0114	0.0685	0.0268	0.0254
GSAGE	348	2,529	0.0170	0.0945	0.0349	0.0569
PE-GSAGE λ = best	222	27,426	0.0114	0.0732	0.0280	0.0464
PE-GQSAGE τ	243	28,481	0.0113	0.0686	0.0266	0.0478
PE-GQSAGE τ, Structure	224	27,385	0.0100	0.0632	0.0248	0.0314
PE-GQSAGE	160	27,417	**0.0089**	**0.0596**	**0.0229**	0.0288
SMACNP	70	748,482	0.0160	0.0881	0.0466	0.1481

As shown in Table 4.2, **PE-GQNN** achieves state-of-the-art performance metrics, with major improvements over traditional GNN, PE-GNN and SMACNP. For the GSAGE layers, PE-GQSAGE achieved the lowest MSE, MAE and MPE. For this type of layer, which gave the best



(a) Test values.

(b) PE-GSAGE predictions.

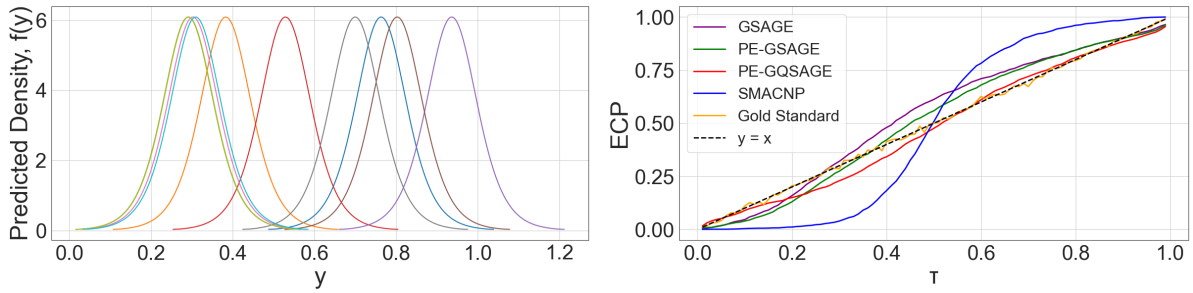
(c) PE-GQSAGE predictions.

(d) SMACNP predictions.

Figure 4.2: Visualization of the predicted results on the California Housing test dataset.

results overall, we still encounter considerable relative improvements from PE-GQSAGE in comparison with PE-GSAGE, with a reduction of 22% in MSE, 19% in MAE, 18% in MPE, and 38% in MADECP. We can also explore in-depth, the contribution of each specific innovation. The τ innovation, which corresponds to the application of the quantile regression framework proposed by Si, Kuleshov, and Bishop (2022), improved the calibration of quantile predictions, reducing MPE and MADECP. The structural innovation, which involves applying the GNN operator only to the features, is instrumental in enhancing prediction performance and improving the calibration, as evidenced by reduced MSE, MAE, MPE and MADECP. Finally, the use of training neighbours' target mean as a feature introduced at one of the last network layers also further improved the model. This last innovation also accelerated convergence during model training, requiring fewer epochs.

Maps of the test values and predictions of selected methods are illustrated in Figure 4.2.



(a) PE-GQSAGE densities.

(b) ECPs for the test set.

Figure 4.3: (a) PE-GQSAGE predicted densities of 10 observations sampled from the California Housing test set. (b) ECP for each τ value used for the California Housing test set.

Visually, the predictions provided by **PE-GQNN** appear to be the closest to the actual data behavior, particularly in the major cities.

Figure 4.3 presents plots that elucidate the behavior of the PE-GQSAGE quantile predictions. Figure 4.3a illustrates the predicted density of a subsample of 10 observations from the test set. For each observation of this sample, the cumulative distribution function was approximated via the quantile predictions using τ values in $[0.001, 0.002, \ldots, 0.999]^\top$. While parametric models presume a rigid structure for their outputs (such as a Gaussian distribution), which constrains their expressiveness, for **PE-GQNN**, no assumptions are made about the form of the predictive distribution. However, as shown in Figure 4.3a, despite the absence of explicit model restrictions, the model produced symmetric distributional shape across predictions, similar to a Gaussian distribution, in this case.

For all test set observations, we verified that no quantile crossings were observed in any of the **PE-GQNN** models, i.e., all predicted quantiles are monotonically increasing with respect to τ, aligning with the expectations described in Section 3.

Lastly, Figure 4.3b displays the empirical cumulative probability (ECP) for the test dataset quantile predictions using each of the 99 τ values in $[0.01, 0.02, \ldots, 0.99]^\top$. This type of plot was proposed by Kuleshov, Fenner, and Ermon (2018). The closer a model gets to the dashed diagonal line, the closer the τ values and the ECP. The Gold Standard represents one Monte

Carlo draw from a perfectly specified model, where for each quantile level, the ECP is the observed success rate in n Bernoulli trials with a success probability of τ, where n is the number of test set instances. It is evident that PE-GQSAGE has by far the best calibration performance. This is particularly notable when compared to SMACNP, which exhibits substantial calibration deficiencies due to its tendency to overestimate the variance component.

4.3 All datasets

Experiments were conducted on two other geographic datasets used by Klemmer, Safir, and Neill (2023) and Bao, Zhang, and Zhang (2024). The Air Temperature dataset (Hooker, Duveiller, and Cescatti (2018)) contains geographical coordinates for $\sim$3,000 meteorological stations worldwide, with the goal of predicting mean temperatures (y) using mean precipitation levels (x). Models were trained with 80% of the data, with 10% for validation and testing each, while SMACNP used a 30% subsample for training, following the specifications of Bao, Zhang, and Zhang (2024). The 3D road dataset (Kaul, Yang, and Jensen (2013)), includes $>$ 430,000 points with latitude, longitude, and altitude for the Jutland, Denmark road network. The task is to interpolate altitude (y) using latitude and longitude ($\boldsymbol{c}$). The data were split into 90% for training, 1% for validation, and 9% for testing. SMACNP metrics are not reported due to high computational costs.

Table 4.3: Performance metrics from three different real-world datasets.

Model	California Housing				Air Temperature				3D road			
	MSE	**MAE**	**MPE**	**MADECP**	**MSE**	**MAE**	**MPE**	**MADECP**	**MSE**	**MAE**	**MPE**	**MADECP**
GCN	0.0222	0.1101	0.0403	0.0475	0.0224	0.1158	0.0427	**0.0334**	0.0170	0.1029	0.0358	0.0560
PE-GCN λ = best	0.0179	0.0935	0.0354	0.0450	0.0045	0.0467	0.0189	0.0640	0.0032	0.0406	0.0151	0.0476
PE-GQCN	0.0114	0.0686	0.0272	0.0262	0.0025	0.0327	**0.0119**	0.0713	**0.0001**	**0.0053**	**0.0022**	0.0439
GAT	0.0227	0.1099	0.0410	0.0586	0.0233	0.1166	0.0434	0.0497	0.0170	0.1030	0.0359	0.0601
PE-GAT λ = best	0.0183	0.0930	0.0352	0.0476	0.0058	0.0566	0.0209	0.0960	0.0035	0.0430	0.0163	0.0551
PE-GQAT	0.0114	0.0685	0.0268	**0.0254**	0.0025	0.0340	0.0143	0.0677	**0.0001**	**0.0053**	**0.0022**	0.0545
GSAGE	0.0170	0.0945	0.0349	0.0569	0.0223	0.1152	0.0431	0.0361	0.0170	0.1031	0.0358	0.0582
PE-GSAGE λ = best	0.0114	0.0732	0.0280	0.0464	0.0037	0.0449	0.0169	0.0720	0.0032	0.0422	0.0146	**0.0417**
PE-GQSAGE	**0.0089**	**0.0596**	**0.0229**	0.0288	0.0023	0.0326	0.0130	0.0785	**0.0001**	0.0054	**0.0022**	0.0786
SMACNP	0.0160	0.0881	0.0466	0.1481	**0.0018**	**0.0290**	0.0391	0.2160	-	-	-	-

Table 4.3 showcases the experimental results obtained from the three datasets: California Housing, Air Temperature, and 3D road. Each GNN layer's performance is evaluated across

three approaches: the traditional GNN, PE-GNN, and **PE-GQNN**. The **PE-GQNN** models incorporate all three innovations discussed in Section 3. Additionally, we include the SMACNP results as a benchmark model based on GPs. Detailed results for the California Housing dataset are also provided in Table 4.2.

As illustrated in Table 4.3, **PE-GQNN** consistently outperforms both traditional GNN and PE-GNN across all datasets and GNN backbones. In every dataset, the **PE-GQNN** innovations lead to significant reductions in MSE, MAE, and MPE. For the California Housing dataset, **PE-GQNN** also achieves the lowest MADECP, indicating superior calibration of quantile predictions.

In the California Housing dataset, **PE-GQNN** consistently outperforms SMACNP in predictive accuracy and provides enhanced uncertainty quantification across all types of GNN layers. Conversely, for the Air Temperature dataset, SMACNP achieves the lowest MSE and MAE but suffers from a major drawback: significantly uncalibrated predictions, reflected by a much higher MPE and MADECP compared to **PE-GQNN**. Consequently, **PE-GQNN** provides the best balance between predictive accuracy and quantile calibration.

Chapter 5

Conclusion

In this work, we have proposed the Positional Encoder Graph Quantile Neural Network (**PE-GQNN**) as an innovative framework to enhance predictive modeling for geographic data. Through a series of rigorous experiments on real-world datasets, we have demonstrated the significant advantages of **PE-GQNN** over traditional GNN, PE-GNN and SMACNP.

Our study highlighted three key innovations within the **PE-GQNN** framework. First, the integration of quantile regression provided inherently calibrated predictions, which were not only more accurate but also offered a comprehensive understanding of the conditional distribution. This approach enhanced the model's ability to capture properties of the conditional distribution without computational overheads, aligning with the principles laid out by Si, Kuleshov, and Bishop (2022). Second, the structural alteration where the GNN operator was applied exclusively to the features, followed by concatenation with the spatial embedding, proved to be instrumental in enhancing prediction performance across various metrics. Lastly, incorporating the mean target variable of the training neighbors as a feature closer to the output layer further improved the model's predictive accuracy by leveraging neighborhood information effectively.

The empirical results underscored the superiority of **PE-GQNN** in achieving lower MSE, MAE, and MPE across all datasets and GNN backbones compared to traditional GNN and PE-GNN. Notably, **PE-GQNN** demonstrated substantial improvements in predictive accuracy and

uncertainty quantification, as evidenced by its consistent performance in quantile calibration metrics such as MPE and MADECP.

In the California Housing dataset, **PE-GQNN** significantly outperformed other models, showcasing its robustness and precision in predictive tasks. For the Air Temperature dataset, while SMACNP achieved the lowest MSE and MAE, it suffered from uncalibrated predictions, which **PE-GQNN** managed to balance more effectively . The 3D road dataset further validated the model's capability in spatial interpolation tasks, where **PE-GQNN** consistently delivered superior performance metrics.

The proposed **PE-GQNN** framework's ability to provide a full description of the predictive conditional distribution, including quantile predictions and prediction intervals, marks a substantial advancement in geospatial machine learning. By leveraging domain-specific innovations and robust experimental validation, this work establishes **PE-GQNN** as a versatile and powerful tool for geographic data analysis. Further research could investigate the integration of **PE-GQNN** with other advanced neural network architectures and explore its potential in dynamic and time-series spatial data contexts.

In conclusion, this work contributes a novel and effective approach to geographic data modeling, providing a solid foundation for future advancements in the field of geospatial machine learning. The promising results and insights derived from this study pave the way for more accurate, reliable, and interpretable predictive models, ultimately enhancing decision-making processes in various geospatial applications. By adhering to the structured methodology and comprehensive evaluation presented in this work, we have laid a robust groundwork for the ongoing evolution and refinement of machine learning models in the area of spatial data analysis.

References

Anselin, Luc (2022). "Spatial econometrics". *Handbook of Spatial Analysis in the Social Sciences*, pp. 101–122.

Appleby, Gabriel, Linfeng Liu, and Li-Ping Liu (2020). "Kriging convolutional networks". *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34(04), pp. 3187–3194.

Bao, Li-Li, Jiangshe Zhang, and Chunxia Zhang (2024). "Spatial multi-attention conditional neural processes". *Neural networks : the official journal of the International Neural Network Society* 173, p. 106201. URL: https://api.semanticscholar.org/CorpusID:268189461.

Bi, Kaifeng et al. (2023). "Accurate medium-range global weather forecasting with 3D neural networks". *Nature* 619.7970, pp. 533–538.

Cressie, N. and C.K. Wikle (2011). *Statistics for Spatio-Temporal Data*. CourseSmart Series. Wiley.

Cressie, Noel, Matthew Sainsbury-Dale, and Andrew Zammit-Mangion (2022). "Basis-Function Models in Spatial Statistics". *Annual Review of Statistics and Its Application* 9, pp. 373–400.

Datta, Abhirup et al. (2016). "Hierarchical nearest-neighbor Gaussian process models for large geostatistical datasets". *Journal of the American Statistical Association* 111.514, pp. 800–812.

Derrow-Pinion, Austin et al. (2021). "Eta prediction with graph neural networks in google maps". *Proceedings of the 30th ACM international conference on information & knowledge management*, pp. 3767–3776.

Fey, Matthias and Jan Eric Lenssen (2019). *Fast Graph Representation Learning with PyTorch Geometric*. arXiv: 1903.02428 [cs.LG].

Guo, Chuan et al. (2017). "On calibration of modern neural networks". *International conference on machine learning*. PMLR, pp. 1321–1330.

Hamilton, Will, Zhitao Ying, and Jure Leskovec (2017). "Inductive Representation Learning on Large Graphs". *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc.

Hooker, Josh, Gregory Duveiller, and Alessandro Cescatti (2018). "A global dataset of air temperature derived from satellite remote sensing and weather stations". *Scientific Data* 5.1, pp. 1–11.

Kashyap, Anirudh Ameya et al. (2022). "Traffic flow prediction models–A review of deep learning techniques". *Cogent Engineering* 9.1, p. 2010510.

Katzfuss, Matthias and Joseph Guinness (2021). "A general framework for Vecchia approximations of Gaussian processes". *Statistical Science* 36.1, pp. 124 –141.

Kaul, Manohar, Bin Yang, and Christian S Jensen (2013). "Building accurate 3d spatial networks to enable next generation intelligent transportation systems". *2013 IEEE 14th International Conference on Mobile Data Management*. Vol. 1. IEEE, pp. 137–146.

Kingma, Diederik and Jimmy Ba (2015). "ADAM: A Method for Stochastic Optimization". *International Conference on Learning Representations (ICLR)*. San Diega, CA, USA.

Kipf, Thomas N. and Max Welling (2017). "Semi-Supervised Classification with Graph Convolutional Networks". *International Conference on Learning Representations (ICLR)*.

Klemmer, Konstantin and Daniel B Neill (2021). "Auxiliary-task learning for geographic data with autoregressive embeddings". *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*, pp. 141–144.

Klemmer, Konstantin, Nathan S Safir, and Daniel B Neill (2023). "Positional encoder graph neural networks for geographic data". *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 1379–1389.

Koenker, Roger and Gilbert Bassett Jr (1978). "Regression quantiles". *Econometrica: journal of the Econometric Society*, pp. 33–50.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". *Advances in neural information processing systems* 25.

Kuleshov, Volodymyr and Shachi Deshpande (2022). "Calibrated and Sharp Uncertainties in Deep Learning via Density Estimation". *ICML*, pp. 11683–11693.

Kuleshov, Volodymyr, Nathan Fenner, and Stefano Ermon (2018). "Accurate uncertainties for deep learning using calibrated regression". *International conference on machine learning*. PMLR, pp. 2796–2804.

Kumar, Ananya, Percy S Liang, and Tengyu Ma (2019). "Verified uncertainty calibration". *Advances in Neural Information Processing Systems* 32.

Lindgren, Finn, Håvard Rue, and Johan Lindström (2011). "An explicit link between Gaussian fields and Gaussian Markov random fields: The stochastic partial differential equation approach". *Journal of the Royal Statistical Society Series B: Statistical Methodology* 73.4, pp. 423–498.

Lv, Yisheng et al. (2014). "Traffic flow prediction with big data: A deep learning approach". *IEEE Transactions on Intelligent Transportation Systems* 16.2, pp. 865–873.

Mai, Gengchen et al. (2020). "Multi-Scale Representation Learning for Spatial Feature Distributions using Grid Cells". *International Conference on Learning Representations*.

Niculescu-Mizil, Alexandru and Rich Caruana (2005). "Predicting good probabilities with supervised learning". *Proceedings of the 22nd international conference on Machine learning*, pp. 625–632.

Pace, R Kelley and Ronald Barry (1997). "Sparse spatial autoregressions". *Statistics & Probability Letters* 33.3, pp. 291–297.

Paszke, Adam et al. (2019). "Pytorch: An imperative style, high-performance deep learning library". *Advances in Neural Information Processing Systems* 32.

Platt, John et al. (1999). “Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods”. *Advances in large margin classifiers* 10.3, pp. 61–74.

Pósfai, Márton and Albert-Laszlo Barabasi (2016). *Network Science*. Citeseer.

Rasmussen, Carl Edward and Christopher K. I. Williams (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Reinhard Furrer, Marc G Genton and Douglas Nychka (2006). “Covariance Tapering for Interpolation of Large Spatial Datasets”. *Journal of Computational and Graphical Statistics* 15.3, pp. 502–523.

Rodrigues, Filipe and Francisco C Pereira (2020). “Beyond expectation: Deep joint mean and quantile regression for spatiotemporal problems”. *IEEE Transactions on Neural Networks and Learning Systems* 31.12, pp. 5377–5389.

Si, Phillip, Volodymyr Kuleshov, and Allan Bishop (2022). “Autoregressive Quantile Flows for Predictive Uncertainty Estimation”. *International Conference on Learning Representations*.

Song, Hao et al. (2019). “Distribution calibration for regression”. *International Conference on Machine Learning*. PMLR, pp. 5897–5906.

Sreenivasa, BR and CR Nirmala (2019). “Hybrid location-centric e-Commerce recommendation model using dynamic behavioral traits of customer”. *Iran Journal of Computer Science* 2.3, pp. 179–188.

Vaswani, Ashish et al. (2017). “Attention is all you need”. *Advances in Neural Information Processing Systems* 30.

Vecchia, A. V. (1998). “Estimation and model identification for continuous spatial processes”. *Journal of the Royal Statistical Society: Series B (Methodological)* 50.2, pp. 297–312.

Veličković, Petar et al. (2018). “Graph attention networks”. *International Conference on Learning Representations*.

Wu, Lingfei et al. (2022). “Graph neural networks: Foundation, frontiers and applications”. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4840–4841.

Xu, Shuai et al. (2020). “Survey on user location prediction based on geo-social networking data”. *World Wide Web* 23.3, pp. 1621–1664.