Title

Deep Learning Methodology and Theoretical Analysis for Change Points, Spatiotemporal and Contaminated Data

Author

GENG, JIALIANG

Publication Date

2025

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Deep Learning Methodology and Theoretical Analysis for Change Points, Spatiotemporal and Contaminated Data

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Statistics

by

Jialiang Geng

2025

ABSTRACT OF THE DISSERTATION

Deep Learning Methodology and Theoretical Analysis for Change Points, Spatiotemporal and Contaminated Data

by

Jialiang Geng

Doctor of Philosophy in Statistics

University of California, Los Angeles, 2025

George Michailidis, Chair

Many modern statistical problems are inherently nonparametric, including a variety of problems in change point detection, graph models, and robust statistics. Previously, the literature is abundant and well studied in parametric and even linear cases in these problems. And for nonlinear cases, many problems remain open and unresolved. In my dissertation, I would like to bridge this gap by leveraging deep learning techniques to solve these nonparametric statistical problems.

Moreover, with the rapid development of deep learning techniques and machine learning theory, we can now conduct asymptotic analysis in a variety of different neural network spaces. And this allows us to thoroughly analyze the convergence rate of the algorithm using these networks.

My first project proposes an offline methodology in change point detection capable of both regression-type problems and multivariate time-evolving data, and provides rigorous theoretical analysis with guarantees for detecting change points and bounding the distance between true and estimated points, covering both independent and dependent sub-Gaussian data sets.

My second project focuses on the graph model and proposes a novel methodology for modeling the time-evolving data, having the capability to predict both existing points with

previous data known and new points with previous data unknown. The proposed model also included an asymptotic analysis of the model convergence rate. Two datasets are used to demonstrate the model performance.

My third project proposes a novel two-step neural network-based model structure combining nuclear norm penalization and output truncation, which is robust to various data contamination mechanisms. The asymptotic analysis on the convergence rate of proposed algorithms is given as well, and the numerical experiments in both synthetic and real datasets are included to demonstrate the model capabilities

In my dissertation, I also include an additional project that I collaborate on with Scientists at the Mayo Clinic. We propose a novel redundancy removal methodology that works on the indexing, archiving, and searching of high-resolution medical images and saves storage and search costs significantly.

The dissertation of Jialiang Geng is approved.

Hongquan Xu

Guang Cheng

Frederic R. Paik Schoenberg

George Michailidis, Committee Chair

University of California, Los Angeles

2025

To my parents and my wife

Contents

List of Figures

List of Tables

Acknowledgements

At the end of my Doctoral journey, I would like to express my gratitude to people who have been encouraging, supporting and guiding me.

First and foremost, I am deeply grateful to my advisor, Professor George Michailidis, for his invaluable mentorship throughout my Ph.D. studies. His profound knowledge, insightful guidance, and unwavering support have been instrumental in shaping my development from a student into an independent researcher. I am especially thankful for his kindness, patience, and respect.

I also wish to sincerely thank the other members of my dissertation committee: Professor Frederic R. Paik Schoenberg, Professor Guang Cheng, and Professor Hongquan Xu for their constructive feedback, thoughtful discussions, and meaningful contributions to my research.

I am fortunate to have shared this journey with my fellow cohort members: Dr. Siwei Ye, Dr. Dehong Xu, Dr. Tianyang Zhao, and Dr. Kaiwen Jiang. It has been a pleasure to learn, grow, and face challenges together. I wish each of them continued success in their future endeavors.

Last but not least, I would like to thank my dear family and friends. Although I was in a hard time during the first few years, you were always with me and showing your support by my side. Finally, I would like to thank Dr. Huilei Wang, who always generously give me emotional support during the hard times. Crossing paths with you is my greatest fortune.

Vita

Education

Nanjing University — Sept. 1st 2015- July. 1st 2019

Bachelor of Science in Statistics

Internships

C3.AI — June 2024 - Sept 2024

Data Science Intern

Mayo Clinic — Sept 2024 - Jan 2025

Data Science Intern

Chapter 1

Introduction

In many fields, including change point detection, graph neural networks, and robust learning, prior research has predominantly focused on linear and parametric models. Recently, there has been growing interest in the asymptotic analysis of neural networks, given their capacity to model complex, nonparametric relationships. However, much of the existing literature on neural networks emphasizes engineering techniques and network architectures, often overlooking theoretical foundations. There remains significant potential for developing robust algorithms in these nonparametric domains, particularly those supported by rigorous theoretical guarantees. For change point detection, plenty of parametric methods have been studied, including exhaustive search [18], penalized cost approaches [44, 52]. Many of these detection algorithms also come with Probabilistic guarantees on the number and locations of the change points detected (asymptotic consistency) for specific parametric models, such as mean shift time series models [34], nonlinear time series models with structural breaks [33], vector autoregressive models [13, 79], regression type models [12, 35], random graph and stochastic block network models [15, 22]. The deep learning based methodology has also been used in change point detection. For instance, [85] has proposed a sequential detection procedure using a predefined detection window and a generalized likelihood ratio test, where the likelihood is approximated by a simple feed-forward neural network. For graph neural

networks, much work has been put in the time series evolved in an interconnected graph structure. Like the change point detection problem, much of the research focuses on parametric and even linear dynamics in temporally evolving mechanisms. Motivated by problems arising in social network analysis, [104] has introduced the network vector autoregression (NAR) model, which captures temporal dependencies in each node's time series not only through its own past values but also through those of its neighboring nodes. There have been selected approaches to extend the above framework to capture nonlinearities. For example, threshold vector autoregressive (TVAR) models [91] generalize their univariate counterparts [89] to the multivariate setting, allowing different linear dynamics to govern the evolution of a vector time series depending on the regime determined by an observed threshold variable. lots of theoretical property developed via learning theory including [68] and [80]. In another stream of literature, the integration of deep learning methodologies with network-based approaches has revolutionized temporal dynamics modeling in recent years. Graph Neural Networks (GNNs) represent a powerful class of neural network architectures specifically designed to process data structured as graphs [97].

And, with the rapid development of neural networks and their learning theory, we are now capable of researching the asymptotic properties and providing the theoretical guarantee for our proposed algorithms in these nonparametric problems. The paper [80] has provided an asymptotic analysis of the feed-forward neural network minimizer

$$R(\hat{f} - f_0) = \mathbb{E}_{f_0}[(\hat{f}(X) - f_0(X))^2] \leq C' \phi_n L \log^2 n$$

where $Y_i = f_0(X_i) + \epsilon_i, \epsilon_i \sim N(0, \sigma^2).i = 1, 2 \ldots n$ and $\hat{f} = \arg\min \sum_{i=1}^{n} (Y_i - f(X_i))^2$. This direction has guaranteed the existence of a minimizer. However, in reality, we usually cannot have strong enough assumptions of the model to guarantee the existence of the global minimizer, and we still want the model to have some kind of asymptotic properties once the loss converges. The paper [68] has proposed a novel graph network MPNN that can be

adjusted to multiple types of graph neural network structures and provides a theoretical guarantee when the loss function is converged. With such innovation in deep learning theory, we can go deep into the asymptotic properties of various algorithms that use neural networks. Recently, we have also seen the recent development of learning theory and analysis in the Huber-like loss function optimization, including [93] and [73], which shed light on the research and development of novel methodologies in algorithms robust to data contamination.

In my dissertation, in chapter 2, we will propose a novel two-step changepoint detection algorithm that uses feed- forward neural networks to detect single and multiple changepoints in multidimensional time-evolving data, as shown in algorithms 1 and 2. In the first step, the neural network is trained over a prespecified window of the data, and its test error function is calibrated over another prespecified window. Then, the test error function is used over a moving window to identify the change point. Once a change point is detected, the procedure involving these two steps is repeated until all change points are identified. It is applicable to both regression-type problems and multivariate time-evolving data. A rigorous theoretical analysis with guarantees for detecting change points and bounding the distance between true and estimated points is included and covering both independent and dependent sub-Gaussian data sets. We compare our algorithms with other widely used algorithms, including KLCPD [20], Double Cusum [25], and Sparsified Binary Segmentation [26]. And our proposed algorithm shows superior performance over those algorithms.

In chapter 3, we will propose a novel methodology for modeling the time-evolving graph structure that leverages both the spatial and temporal information. The spatial and temporal information is modeled by separate graph neural networks and combined by addition. And most importantly, the algorithm can predict the node response for both existing nodes (nodes whose node features and node response before current time point t are available) and new unknown nodes (whose previous node features and node response are not available) at each time point t. We provide the asymptotic analysis theorem that derives the convergence rate of the algorithm. We demonstrate the performance of our algorithm on two real datasets,

Miami housing and AQI data in the cities of China, and show the performance advantage over other existing methodologies.

In chapter 4, we will propose a two step neural network-based algorithm that combines the nuclear norm penalization, which encourages a low-rank neural network structure, and a series of neural networks with output truncation to get a neural network estimator robust to data contamination. The nuclear norm penalization generates a low-rank estimator that captures the main part of the model, and truncated neural networks are further used to fill the gap between the low-rank estimator and the target. We also provide the asymptotic analysis of the convergence rate. The proof of the theorem also uses a similar β Hölder smoothness property in chapter 2 and 3. We also evaluate the algorithms on both synthetic data and real datasets for both regression and classification scenarios.

In chapter 5, we will introduce the research that is collaborated with the researchers at the Mayo Clinic: Dr. Hamid R. Tizhoosh and Dr. Saghir Alfasly. And we will propose the ARReST (Antagonistic Redundancy Reduction Strategy), an oppositional approach that leverages redundancy among dissimilar classes to minimize the number of image patches that need to be indexed. ARReST optimizes storage, reduces computational costs, and accelerates image search, making retrieval more efficient, which will alleviate the high cost of high-performance storage problems in digital pathology.

Chapter 2

Neural network based change point detection algorithm

2.1 Introduction

In this chapter, we will propose a two-step offline change point detection algorithm and provide asymptotic analysis of the accuracy of the detected change points.

The problem of change point detection has a rich history across various disciplines, including statistics, econometrics, signal processing, and machine learning, owing to its wide range of applications in areas such as quality control [55], economics, finance, and risk analysis [103], medical diagnosis [60], social network analysis [51], and linguistics [54, 95]. Change point detection methods are typically categorized according to the task in hand: *online* methods aim to detect changes in the data distribution/dynamics, as soon as they occur based on streaming data, while *offline* methods aim to *retrospectively* detect changes, when all data are available for analysis. The former task is also referred to in the literature as event or anomaly detection, while the latter task is also sometimes called signal segmentation.

Various detection algorithms have been developed for a number of *parametric* models in the offline setting, including exhaustive search [18], penalized cost approaches [44, 52], and

approaches based on screening and ranking criteria [70]. Many of these detection algorithms also come with probabilistic guarantees on the number and locations of the change points detected (asymptotic consistency) for specific parametric models, such as mean shift time series models [34], nonlinear time series models with structural breaks [33], vector autoregressive models [13, 79], regression type models [12, 35], random graph and stochastic block network models [15, 22]. Meanwhile, evaluating different change point detection algorithms is equally important. The paper by Truong et al. [90] provides a comprehensive overview of offline change point detection methods for multivariate time series, discussing both consistency properties—such as asymptotic consistency—and evaluation metrics, including the Hausdorff distance and F1 score, many of which will also be employed in this work.

The rapid advancement and widespread adoption of neural network architectures have underscored the need for change point detection methods specifically tailored to these models, as most existing approaches are designed for parametric settings. However, the current literature in this area remains limited and has primarily focused on the online change point detection problem. For instance, [85] has proposed a sequential detection procedure using a predefined detection window and a generalized likelihood ratio test, where the likelihood is approximated by a simple feed-forward neural network. Similarly, [57] has reformulated the detection problem as a classification task, employing neural network architectures to train a classifier. In [29], change point detection is carried out using a novel deep neural network (DNN) structure called the Pyramid Recurrent Neural Network, which integrates recurrent neural networks (RNNs) and convolutional neural networks (CNNs); the output is then passed through a binary classifier to determine the presence of a change point. Another approach, presented in [19], combines singular spectral analysis with auto-associative neural networks, using residuals and a specified threshold to detect change points. Lastly, [38] incorporates preprocessing steps such as normalization and recursive singular spectral analysis with a neural network-based autoencoder, where the reconstruction error serves as the detection criterion under a predefined threshold.

On the other hand, to the best of our knowledge, change point detection methods in the nonparametric offline setting for time-evolving data have primarily focused on detecting distributional changes, rather than shifts in the underlying time-evolving mechanisms. For example, [9] has proposed a kernel-based model selection framework that minimizes a penalized kernel least-squares criterion, enabling the detection of an unknown number of change points. Similarly, the method in [105] introduces a robust offline time segmentation approach by combining dynamic programming, cost-based optimization, and also provides theoretical error guarantees. The work in [62] has proposed a nonparametric approach that leverages machine learning classifiers, such as random forests, to construct a classifier-based log-likelihood ratio for detecting multiple change points in time series data. The paper [2] has proposed a Cumulative-Sum (CUSUM) algorithm for detecting change points in multivariate time series, but focusing on changes in the spectral characteristics of the data, rather than the more common mean or variance shifts.

However, deep learning-based methodologies remain relatively sparse when applied to the offline change point problem. The work in [20] has used a deep kernel parameterization and an auxiliary generative model to optimize the power of the corresponding test statistic, enabling effective detection of diverse change types even in the presence of limited samples. [58] has proposed a data-driven framework for offline change point detection by training neural networks to automatically approximate popular testing procedures and provided theoretical guarantees on their error rates.

These papers primarily focus on detecting changes in the data distribution rather than shifts in the underlying time-evolving mechanisms, and many lack theoretical analysis regarding asymptotic consistency. Further, some of them –e.g., [2, 62]– only address the single change point problem. Therefore, the goal of this work is to develop a detection algorithm that leverages feed-forward networks to identify changes in time-evolving data. The main contributions are: (i) the development of a nonparametric framework leveraging feed-forward neural networks to detect single and multiple change points in multidimensional time-evolving

data, as shown in algorithms 1 and 2. (ii) applicability to both regression-type problems and multivariate time-evolving data, (iii) a rigorous theoretical analysis (Section 2.4), with guarantees for detecting change points and bounding the distance between true and estimated points, covering both independent and dependent sub-Gaussian data sets.

The remainder of the paper is organized as follows: Section 2.2 sets up the problem under consideration and introduces the nonparametric criterion function. Section 2.3 describes the two-step algorithm, while Section 2.4 provides the theoretical guarantees for it. Finally, Section 2.5 presents numerical experiments based on simulated data that showcase the role that various assumptions play in the performance of the algorithm, while Section 2.5 illustrates the algorithm on selected real-world data sets.

2.2 Problem Formulation

The setting under consideration is as follows: Let $Y_i, i = 1, \cdots, T_{\text{sum}}$ be a response/output vector of dimension h and $X_i, i = 1, \cdots, T_{\text{sum}}$ a vector of dimension p of covariates/attributes, related through the following predictive model:

$$Y_i = f_j(X_i) + \epsilon_i, \quad \tau_{j-1} < i \leq \tau_j \tag{2.1}$$

where $\tau_0 \leq i \leq \tau_{N+1}, \tau_0 = 0, \tau_{N+1} = T_{\text{sum}}$ correspond to the N *change points* and T_{sum} is the number of total time points available. It can be seen that the model f_j is applicable for a segment of time points and changes to another one f_{j+1} at the change point $\tau_j + 1$. The model function $f_j, \mathbb{R}^p \to \mathbb{R}^h$ is assumed to be measurable and composed of several functions (technical details provided in the Supplement). This framework can also cover non-linear multivariate autoregressive time series models. For illustration purposes, consider the linear Vector Autoregressive (VAR) model:

$$Y_i = A_{1j} Y_{i-1} + ... + A_{qj} Y_{i-q} + \epsilon_i, \tau_{j-1} \leq i \leq \tau_j \tag{2.2}$$

where $Y_i \in \mathbb{R}^h$ and $A_{ij} \in \mathbb{R}^{h\times h}, 1 \leq i \leq q$ is the transition matrix. Considering the q lags of Y_i as $X_i = Y_{i-q\leq k<i}$, we get $Y_i = \bar{f}_j(X_i) + \epsilon_i$ where $\bar{f}_j = \bar{f}_j(A_{ij})$ which can be covered by the proposed framework (additional technical details provided in Section 2.5).

Criterion function: The following criterion function is used in the detection algorithm: $E(t) = \sum_{i=t}^{T_2+t} ||Y_i - \hat{f}(X_i)||^2$, where $\hat{f} : \mathbb{R}^{p_0} \to \mathbb{R}^{p_{L+1}}$ corresponds to a fully connected feed-forward network defined as: $\hat{f} = W_L\sigma_{v_L}W_{L-1}\sigma_{v_{L-1}}...W_1\sigma_{v_1}W_0x$, with L being the number of layers of the network, W_j the $p_{j+1} \times p_j$ weight matrix for layer $j = 1, \cdots, L$, $v_j \in \mathbb{R}^{p_j}$ the bias vector and $\sigma_{v_j}(x) = \max(0, x - v_j)$ the ReLU activation function. Note that $p_0 = p$ and $p_{L+1} = h$, while $||.||$ denotes the ℓ_2 vector norm. The criterion function measures the squared error loss between the actual and fitted values and mimics analogous criteria used in change point analysis for traditional parametric statistical models. Further, the neural network is trained over a certain time period (to be specified in section 2.3), and then the criterion function is evaluated over a window of size T_2, whose starting point is a generic time point t. The length of the time window T_2 is also specified in Section 2.3.

2.3 Detection Algorithm

To build intuition and clarify the core ideas, we begin by presenting the detection algorithm in the simpler setting of a *single change point*—a problem that, while more tractable, has garnered significant attention in the literature (see, e.g., [12, 28]). This case serves as a stepping stone toward understanding and extending the methodology to the more realistic and challenging scenario of *multiple change points*.

The proposed detection algorithm proceeds in two main steps. First, we train a feedforward neural network and evaluate its test error across the time index to construct an error-based criterion function $E(t)$, defined over $t = 1, \ldots, T_{\text{sum}}$, as described in Algorithm 1. In the second step, we compare the values of $E(t)$ against a pre-specified threshold to determine potential change points, which are then identified at time points where the criterion exceeds

the threshold. This decision step is outlined in Algorithm 2.

2.3.1 Single Change point detection

Suppose that the signal is generated according to the following mechanism

$$y_i = f_0(x_i)I(i < \tau) + f_1(x_i)I(i \geq \tau) + \epsilon_i, i = 1, \cdots, T_{\text{sum}}, \tag{2.3}$$

where $x_i \in \mathbb{R}$, $f_0, f_1 : \mathbb{R}^p \to \mathbb{R}^1$ are the signal generation functions, ϵ_i is mean zero, finite variance iid noise, T_{sum} is the the total number of time points observed and $\tau \in \{1, \cdots, T_{\text{sum}}\}$ denotes the change point. It can be seen that at τ the data-generating mechanism switches from f_0 to f_1. The generated data based on the f_0 and f_1 mechanisms are shown in Figure 2.1, comprising $T_{\text{sum}} = 1000$ total observations, with the true change point located at $\tau = 500$. Suppose that a feed-forward neural network is trained on an interval of length T_1 (a tuning parameter) with training window size, while the error criterion function is evaluated over a test window of length T_2. If the error criterion function $E(t), t \in ((j-1)T_2 + 1 : jT_2, j = 1, \cdots, \lceil T_{\text{sum}}/T_2 \rceil$ does not exceed a prespecified threshold (specified in the sequel), then the window is moved and the error criterion function computed again, until a change point is identified. When a change point is identified then a new feed-forward neural network is trained and the process repeated. The parameters T_1 and T_2 are selected by the user. In the illustrative example in Figure 2.1, $T_1 = 100$ and $T_2 = 100$. It can be seen that the error remains low in the interval $(100, 500)$, as indicated by selective evaluations of $E(t)$ (yellow stars), while it jumps in the $(500, 600)$ window. Then, a new model is trained, which leads to a reduction in the values of the $E(t)$ function.

The first step of the algorithm generating the error function $E(t)$ is presented in algorithm 1. Given a time point t, let $X[t - T_1 : t], Y[t - T_1 : t]$ be the input and the output signal from time $t - T_1$ to t, of dimension $T_1 \times p, T_1 \times h$, respectively. A feed forward network $\hat{f}_\theta$ is fitted for this input-output pair and the citerion function $E(t)$ is calculated by the test error for

time window $(t, t+T_2)$. The detection step for the single change point case is rather simple;

Algorithm 1 Test Error Generator

1: **System Initialization**
2: Input data X_t, prediction $Y_t = f_i(X_t) + \epsilon_t,\ t = 1, 2, \ldots$, training window size T_1, test window size T_2, maximum training step T, neural network function $\hat{f}_\theta$ with trainable parameter θ.
3: **for** $t = T_1, 2, \ldots, T$ **do**
4: Compute the loss:

$$\mathcal{L}_\theta = \|Y[t - T_1 : t] - \hat{f}_\theta(X[t - T_1 : t])\|^2$$

5: **while** θ not converged **do**
6: Update parameters: $\theta \leftarrow \text{ADAM}(\mathcal{L}_\theta)$
7: **end while**
8: Record test error:

$$E[t] = \|Y[t : t + T_2] - \hat{f}_\theta(X[t : t + T_2])\|^2$$

9: **end for**
10: **Output** test error function $E(t)$

namely, select the time point t that maximizes the function $E(t)$ as the estimated change point. Specifically, $\tau = \arg\min E(t), t = T_1 : +1, \cdots, T_{\text{sum}}$. In the illustrative example, we have that $\tau = 500$. Probabilistic guarantees for the accuracy of the location of the identified change point are provided in Theorem 1.

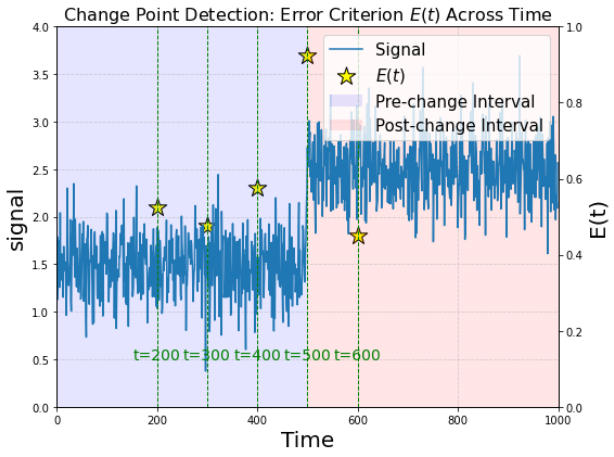


Figure 2.1: Example for generating E(t)

2.3.2 Multiple Change points

Detection becomes more challenging with an unknown number of change points. The choice of window size is crucial—if multiple change points fall within the same training or testing window, some may be missed. To address this, we introduce a detection window with a tunable size parameter T_3, and a detection threshold ϕ to distinguish between windows that contain a change point and those that do not. Strategies for selecting the tuning parameters T_3 and ϕ are detailed in Sections 2.3.3 and 2.4.

In the modified detection step, for each time point t, we define a detection window $(t - T_3, t + T_3)$ centered at t, and compute $E(t)$ within this window. If the difference between the maximum and minimum exceeds a pre-specified threshold ϕ, a change point is detected at the time that maximizes $E(t)$ within the window. The detection window then shifts to a non-overlapping region. Theoretical guarantees for the accuracy of the estimated change point, as well as guidance for selecting window sizes and thresholds, are provided in Section 2.4.

For example, consider a test error curve with a single peak at time point 50 in Figure 2.2, with a peak threshold $\phi = 4$, and fix the detection window size $T_3 = 15$. Before time $t = 44$, the range of the error function within the detection window remains below ϕ, as illustrated by the first blue arrow, so the window advances. Between time points 44 and 54, near the peak, the range exceeds the threshold (second blue arrow), and the time point maximizing the test error within the window is recorded as a change point. After detecting a change point, the detection window jumps forward by its full width to avoid overlapping with previous windows, thereby preventing repeated detections of the same change point. This strategy is further discussed in the next section.

The detailed detection strategy is described in algorithm 2

Algorithm 2 Multiple Change Point Detection

System Initialization
Input error function $E(t)$, detection window size T_3, $n = 1$, detection threshold ϕ, total time points T_{sum}.
for $t = 1, 2, \ldots, T_{\text{sum}}$ **do**
  Calculate

$$c_t = \max_{\max(t-T_3,0)\le i,j\le \min(t+T_3,T_{\text{sum}})} |E(i) - E(j)|$$

  if $c_t \ge \phi$ **then**
    $\hat{\tau}_n = \arg\max_{\max(t-T_3,0)\le i,j\le \min(t+T_3,T_{\text{sum}})} E(i)$
    $n = n + 1$
    $t = t + 3T_3$
  end if
end for
Output: $\hat{\tau}_1, \ldots, \hat{\tau}_N$.

2.3.3 Selection of the window size and detection threshold

Given the model defined in equation (2.1) with total time points T_{sum}, the single change point case is straightforward, as we can simply select $t = \arg\max_t E(t)$. However, in the case of multiple change points, selecting appropriate window sizes for training, testing, and detection becomes crucial.

To simplify the process, we set the training, testing, and detection window sizes as $T_1 = T_2 = T_0$ and $T_3 = 2T_0$ for some base window size T_0. Similar to standard assumptions in the literature ensuring that the true change points have sufficient distance from the boundaries of the observation period in the data, we also require that the change points are not too densely distributed over the time horizon T_{sum}. Specifically, there should exist a minimum separation between change points to avoid overlap within a single detection window and to ensure sufficient data for the neural network to train effectively.

Under this setting, the window size T_0 governs the maximum number of change points that can be reliably detected, with the upper bound $N \le \frac{T_{\text{sum}}}{T_0}$. Further details are provided in Theorems 2

For independent and identically distributed data with finite variance, a good choice is

$T_0 = \sqrt{T_{\text{sum}}}$. For data sets with higher-order moments or sub-Gaussian distributions, a smaller window size may suffice. In contrast, for time-dependent data, a larger window is typically required (see Section 2.4).

Additionally, in the multiple change point setting, we require a sufficiently large detection threshold ϕ to distinguish windows that contain change points from those that do not. The threshold should be smaller than the minimal model signal strength $M1_*$, and we use $\phi = \frac{1}{3}M1_*$ in our implementation. This implies another key assumption: the signal strength $M1_*$ must be sufficiently large to be distinguishable from random fluctuations in the absence of change points. The precise configuration is discussed in Theorem 2 in Section 2.4.

2.4 Theoretical Analysis of the Detection Algorithm

Recall that T_{sum} denotes the total number of time points available, as defined in Section 2.2. The parameters T_1, T_2, and T_3 refer to the training window size, testing window size, and detection window size, respectively. The symbol ϕ denotes the detection threshold. We further assume that there are $N+1$ total change points, with $\tau_0 = 0$ and $\tau_{N+1} = T_{\text{sum}}$.

The following assumptions are imposed to establish probabilistic guarantees for the proposed detection algorithm.

Assumption 1 (Signal Strength)**.** The change in the signal strength of the model is bounded above and below, as

$$M1_i \leq \mathbb{E}\left[\|f_{j+1}(X) - f_j(X)\|^2\right] \leq M2_i < \infty,\ j = 0, \cdots, N$$

where f_i is the model function defined in Equation 2.1, and $M1_i, M2_i$ is some constant. Define $M1_* = \min_i M1_i$ as the minimum model change signal. We further assume:

$$M1_* > 4h\sigma^2,$$

where σ^2 denotes the variance of the noise, and $h > 0$ is a constant.

This assumption ensures that the signal change between successive models is sufficiently large compared to the noise level, allowing the detection algorithm to reliably identify change points.

Assumption 2 (Minimum Distance Between Change Points)**.** Define $T = \min_{i=0,\ldots,N} |\tau_{i+1} - \tau_i|$ as the minimum distance between consecutive change points. We assume there exists a constant C_0 and function b such that

$$T \geq \frac{C_0 b(T_{\text{sum}})}{M 1_*},$$

where typically $b(T_{\text{sum}}) = \sqrt{T_{\text{sum}}}$.

This assumption provides a lower bound on the separation between true change points, balancing the distance and the magnitude of model change. Such trade-offs between signal strength and spacing are standard in the literature (see, e.g., the review paper [90]). The assumption also implicitly requires change points to be sufficiently far from the boundaries, which enables the model to be *pretrained* before the first change point and retrained after any detected changes. This is a common requirement in change point detection analysis.

Assumption 3 (Model Smoothness)**.** Each model function f_j can be written as a composition $f_j = g_q \circ g_{q-1} \circ \cdots \circ g_1 \circ g_0$, where each component g_i is a β-Hölder smooth function.

This assumption guarantees the smoothness of the model function, which ensures that the neural network can accurately approximate the true model. A detailed definition of β-Hölder smooth functions is provided in the supplementary Section A.1.1.

The next result establishes guarantees regarding the performance of the proposed detection algorithm for a single change point, which is of independent interest.

Theorem 1. *Consider a single change point cases based on the model in equation 2.1 and the error criterion function* $E(t) = \sum_{i=t}^{t+T_0} ||\hat{f}(X_i) - Y_i||^2$. *Recall that we set* T_0 *as the training*

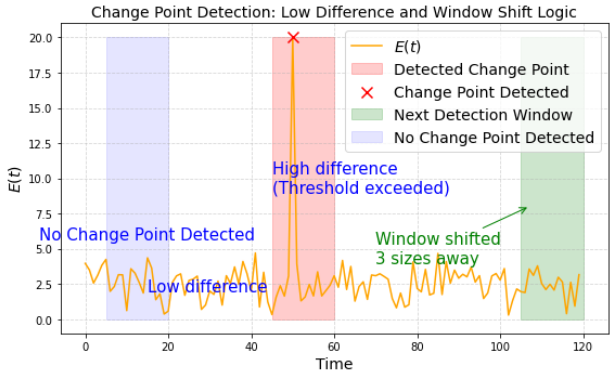


Figure 2.2: Illustration of the detection process based on the error criterion $E(t)$

sample size and test size with the neural network $\hat{f}$ *trained on* $t - T_0 \leq i \leq t$. *Then, there exists a* t_0, *such that* $E(t_0) - E(t) > \phi$ *defined in Section 2.3 for all* t.

Further, let $\hat{\tau} = \arg\max_t E(t)$ *with* τ *denoting the true change point. Then, there exists a constant* C, *such that*

$$|\hat{\tau} - \tau| \leq \frac{CT_0}{M_1 - 2h\sigma^2}, \tag{2.4}$$

with probability $1 - \frac{C_l}{T_0^{l-1}}$, *for some constant* C_l *depending on the number of moments of* $f(\cdot)$*–i.e.,* $\mathbb{E}[||f_i(X)||^l] < \infty, l \geq 2$*– and with probability* $1 - C_1 e^{-C_2 T_0}$ *for some constants* C_1 *and* C_2, *if the output of functions* $f_j(X), \hat{f}(X)$ *is sub-Gaussian based on input data* X.

Theorem 1 establishes that Step 1 of the detection algorithm will identify a detectable peak in the error function near the true change point τ. Moreover, the estimated change point $\hat{\tau}$, detected by the algorithm, will be sufficiently close to the true change point. As a result, once a change point is detected, the subsequent detection window—shifted by one full detection window size—will not overlap with the previously detected change point. This ensures that each true change point is detected only once by the algorithm.

Next, we consider the case of multiple change points and provide probabilistic guarantees for the proposed detection algorithm regarding the estimated number and locations of them.

Theorem 2. *Consider input data* $X_t, t = 1, \cdots, T_{sum}$ *with the output signal* y_t *generated according to model in equation* (2.1). *Further, assume that assumptions 1, 2 and 3 hold.*

Case 1: Let $\mathbb{E}[||f_i(X)||^l] < \infty$ *for* $l > 2$*, select a constant* $\rho \in (0,1)$ *and define the function in assumption 2 as* $b(T_{sum}) = \sqrt[\rho(l-1)]{T_{sum}}$*. Then, assuming that the data are iid, the combination of Algorithms 1 and 2 with* $T_0 = T^\rho = (\frac{C_0}{M1_*})^{\rho\ l-1}\sqrt{T_{sum}}$ *for large* l*, detection window size* $2T_0$ *and detection threshold described in section 2.3.3, yields as* $T_{sum} \to \infty$*,*

$$\mathbb{P}\left(\hat{N} = N, \max_{i=1,\dots,N} M1_*|\tau_i - \hat{\tau}_i| < C\frac{M1_*}{M1_* - 2h\sigma^2}T_{sum}^{\frac{1}{l-1}}\right) \to 1 \tag{2.5}$$

Case II: Assume iid input data, and that the output of model function f_j *and neural network* $\hat{f}$ *is sub-Gaussian. Further, define the function in assumption 2 as* $b(T_{sum}) = \sqrt[\varrho]{log(T_{sum})}$*. Then, the combination of algorithms 1 and 2 with* $T_0 = T^\rho = (\frac{C_0}{M1_*})^\rho \log(T_{sum})$*, detection window size* $2T_0$ *and detection threshold described in section 2.3.3, yields as* $T_{sum} \to \infty$*.*

$$P\left(\hat{N} = N, \max_{i=1,\dots,N} M1_*|\tau_i - \hat{\tau}_i| < C\frac{M1_*}{M1_* - 2h\sigma^2}\log(T_{sum})\right) \to 1 \tag{2.6}$$

Case III: Assume that the input data are dependent and in addition, the following condition holds: $|Cov(f_i(X), f_j(X))| < c < 1, i \neq j$ *for some constant* c*. Further, assume that* $\mathbb{E}[||f_i(X)||^2] < \infty$*. Select two constants* $\rho, \psi \in (0,1)$ *and define function in assumption 2* $b(T_{sum}) = \sqrt[\psi\rho+1]{T_{sum}}$*. Then, the combination of Algorithms 1 and 2 with* $T_0 = T^\rho = (\frac{C_0}{M1_*})^\rho (T_{sum})^{\frac{\rho}{\psi\rho+1}}$*, detection window size* $2T_0$ *and detection threshold described in section 2.3.3, yields as* $T_{sum} \to \infty$*.*

$$P\left(\hat{N} = N, \max_{i=1,\dots,N} M1_*|\tau_i - \hat{\tau}_i| < C\frac{M1_*}{M1_* - 2h\sigma^2}T_{sum}^{\frac{\rho}{1+\psi\rho}}\right) \to 1 \tag{2.7}$$

Remark: The theorem establishes that, across different data regimes—i.i.d., sub-Gaussian, and dependent—with high probability, the number of detected change points matches the number of true change points. Furthermore, the distances between the estimated and true change points are bounded with high probability. A key assumption underlying the theorem

is the existence of a lower bound on the spacing between consecutive change points. This spacing condition is crucial for ensuring detectability of the change points and also determines the maximum number of change points that can be reliably detected, given the total number of observed time points T_{sum}. In the case of sub-Gaussian data, the theorem additionally provides a tighter error bound on the distance between estimated and true change points. It also enables the detection of a greater number of (potentially denser) change points within the same observation window. This improvement leverages the concentration properties of sub-Gaussian input data to strengthen the theoretical guarantees and enhance the performance of the detection algorithm. Finally, for dependent data, the rate for localizing change points is inferior than that of independent data, since the dependence impacts the "effective" sample size available in the analysis.

2.4.1 Remarks

The different assumptions in Theorems 2, impose different lower bounds on the minimum distances between true change points, depending on the characteristics of the data set.

For independent data sets with finite l-th moment of the model output (for sufficiently large l), the minimum spacing between consecutive change points can scale as $T_{\text{sum}}^{\frac{1}{\rho(l-1)}}$. Consequently, the number of detectable change points scales as $T_{\text{sum}}^{1-\frac{1}{\rho(l-1)}}$ for a fixed total number of time points T_{sum}.

For sub-Gaussian data sets, the required minimum spacing can be as small as $\log T_{\text{sum}}$, allowing the detection of a greater number of change points for the same T_{sum}.

In contrast, for temporally dependent data, the lower bound on the minimum spacing is at least $T_{\text{sum}}^{1/2}$, thereby limiting the number of detectable change points to at most $\sqrt{T_{\text{sum}}}$.

2.5 Performance Evaluation Experiments

In this section, we present several experiments to illustrate the performance of the proposed algorithm and examine the sensitivity of the results based on the posited assumptions underlying the theoretical results.

2.5.1 Synthetic Data Experiments

We focus on experiments with synthetic data and consider the following three settings: the first two involve i.i.d. responses and predictors (Y_i, X_i) generated by different nonlinear mechanisms, while the third setting involves time series data, where the predictors X_i correspond to past lags of the response variables.

Following the regression model in Equation (2.1), the input data X_i is generated from an i.i.d. normal distribution $\mathcal{N}(0, I)$, where $X_i \in \mathbb{R}^{400}$, and the output $Y_i \in \mathbb{R}^{200}$. The noise term ϵ_i is sampled i.i.d. from $\mathcal{N}(0, \sigma^2 I)$, with $\sigma = 0.4$ and I denoting the identity matrix.

Feed-forward network generated data Each model f_j is implemented as a three-layer feed-forward neural network. In each layer, the fully connected weight matrix is given by $W_i + W_{ij}$, with no bias term, and is followed by a ReLU activation function. Here, W_i is a fixed matrix shared across all model functions for layer i, while W_{ij} is a sparse matrix unique to model f_j at layer $i = 1, 2, 3$. The dimensions of the weight matrices are as follows: $W_1, W_{1j} \in \mathbb{R}^{400\times100}$, $W_2, W_{2j} \in \mathbb{R}^{100\times100}$, and $W_3, W_{3j} \in \mathbb{R}^{100\times200}$. The entries of W_i are drawn from $\mathcal{N}(0, 1)$, while entries of W_{ij} follow a sparse distribution defined as $\text{Sparse}(0.1) \cdot \mathcal{N}(0, 1)$, where each entry is nonzero with probability 0.1 and zero otherwise.

Random forest generated data Each model f_j is trained as a separate random forest regressor mapping from $\mathbb{R}^{400}$ to $\mathbb{R}^{200}$. To generate f_j, we first define a nonlinear function $g : \mathbb{R}^{400} \to \mathbb{R}^{200}$ (constructed using the feed-forward network described above with weights $W_i + W_{ij}$). Then, for each input X_i, we compute $\tilde{Y}_i = g(X_i)$, and train the random forest f_j

by minimizing the squared loss: $f_j = \arg\min_f \|f(X_i) - \tilde{Y}_i\|^2$. The final response is given by $Y_i = f_j(X_i)$.

Mutlivariate time series autoregressive data The third data set is generated from a Vector Autoregression (VAR) model with q lags, given by:

$$Y_t = A_{1j}Y_{t-1} + \cdots + A_{qj}Y_{t-q} + \epsilon_t, \quad \tau_{j-1} \leq t \leq \tau_j, \tag{2.8}$$

where each $Y_t \in \mathbb{R}^{200}$, and $A_{ij} \in \mathbb{R}^{200\times 200}$ for $1 \leq i \leq q$ are the transition (regression coefficient) matrices.

This model can be reformulated as a special case of the nonlinear regression model in Equation (2.1). By flattening the predictor as $X_t = (Y_t, Y_{t-1}, \ldots, Y_{t-q+1})$ and defining the extended noise vector $\boldsymbol{\epsilon}_t = (\epsilon_t, \ldots, \epsilon_{t-q+1})$, we can write:

$$Z_t = B_j Z_{t-1} + \boldsymbol{\epsilon}_t,$$

where

$$B_j = \begin{bmatrix} A_{1j} & A_{2j} & \cdots & A_{qj} \\ I & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & I & 0 \end{bmatrix}, \tag{2.9}$$

and I is the 200×200 identity matrix. Here, different models f_j are represented by different matrices B_j, and $B_j Z_{t-1} = f_j(Z_{t-1})$. Thus, this flattened VAR(q) model is a specific instance of the general regression framework with dependent inputs, and its consistency is covered under Theorem 2.

In our experiments, we use a VAR(4) model. The coefficient matrices for the different models are generated according to the procedure described in [4].

Table 2.1 reports the performance of the proposed algorithm on the baseline simulation

Table 2.1: Results on synthetic data experiments

Model	Distance Between CPs	N	$\bar{d}_{\tau-\hat{\tau}}$	$\bar{d}_{\lvert N-\hat{N}\rvert}$	Proportion ($\hat{N}=N$)
Feed-Forward Network Data	300–1500	30	17.68	0.0	1.0
Random Forest	300–1500	30	17.30	0.0	1.0
VAR(4) Flatten	900–1500	30	20.89	0.0	1.0

experiments. The neural network employed in the training step is a fully connected feedforward one with two hidden layers, each comprising 256 units and ReLU activation functions. The change points τ_j are generated randomly. The "distance between change points" refers to the range used by the random generator to determine the *separation between successive change points*, as required by Assumption 2. The parameter N denotes the number of change points.

For each combination of distance range and number of change points N, we generate 10 synthetic datasets and apply the proposed two-step algorithm. The performance is evaluated using the following metrics:

$$\bar{d}_{\tau-\hat{\tau}} = \frac{1}{10}\sum_{i=1}^{10} d_i(\tau-\hat{\tau}), \quad \text{where} \quad d_i(\tau-\hat{\tau}) = \frac{1}{N}\sum_{j=1}^{N}|\tau_j-\hat{\tau}_j|,$$

with τ_j and $\hat{\tau}_j$ representing the jth true and estimated change points, respectively. The estimated change point $\hat{\tau}_j$ is chosen as the closest detected point to τ_j Further, we report

$$\bar{d}_{|N-\hat{N}|} = \frac{1}{10}\sum_{i=1}^{10}|N-\hat{N}_i|,$$

where $\hat{N}_i$ is the number of change points estimated in the ith repetition, and $\text{Prop}(\hat{N}=N)$ is the proportion of the 10 repetitions for which $\hat{N}_i = N$.

The next set of simulation experiments illustrates the accuracy of the detected change point locations and the number of detected change points across different types of data sets. The first comparison focuses on independent versus dependent input data.

In the independent setting, the inputs are generated from an i.i.d. normal distribution

Table 2.2: Independent vs dependent input

model	Distance between cps	N	$\bar{d}_{\tau-\hat{\tau}}$	$\bar{d}_{\lvert N-\hat{N}\rvert}$	Prop($\hat{N}=N$)
Independent input data	200-300	30	18.76	0.3	0.7
Dependent input data	200-300	30	88.14	7.7	0.0

$\mathcal{N}(0, I)$, while in the dependent setting, the inputs are generated using a VAR(1) model. To ensure a fair comparison, both types of inputs are normalized to the same scale for each X_i. The model function f_j in both cases is the same feed-forward neural network architecture used in the previous set of experiments.

To facilitate the comparison, we consider a relatively narrow range for the distance between change points, set between 200 and 300 time points.

The results in Table 2.2 show that the detection algorithm achieves better performance—measured by a smaller average distance to true change points and a higher proportion of correctly detected change point counts—when the input data are independent rather than dependent. This observation aligns with the theoretical results established in Theorem 2.

Further, Theorem 2 highlights how the distributional properties of the model outputs influence algorithmic performance. Table 2.3 presents presents a comparison between sub-Gaussian and non-sub-Gaussian model outputs. For both data sets, the model function f_j is implemented using the same feed-forward neural network structure. The weight matrices W_i in each layer are generated from a normal distribution $\mathcal{N}(0, 1)$ across both settings. However, the perturbation matrices W_{ij} differ in their distribution: for the sub-Gaussian case, entries are generated as Sparse(0.1) $\times\, \mathcal{N}(0, 1)$, while for the non-sub-Gaussian case, entries follow Sparse(0.1) $\times$ Uniform$(0, 1)$. To ensure a fair comparison, we fix the model change signal across both settings. Specifically, we enforce that for all $i = 1, 2, \ldots, N$,

$$\mathbb{E}_{p(X)}\left[\|f_i(X) - f_{i-1}(X)\|^2\right] = 50.$$

Table 2.3 shows that the detection algorithm achieves better performance when the model

outputs are sub-Gaussian, aligning with the theoretical guarantee obtain in Theorem 2.

Table 2.3: Non sub-Gaussian vs sub-Gaussian

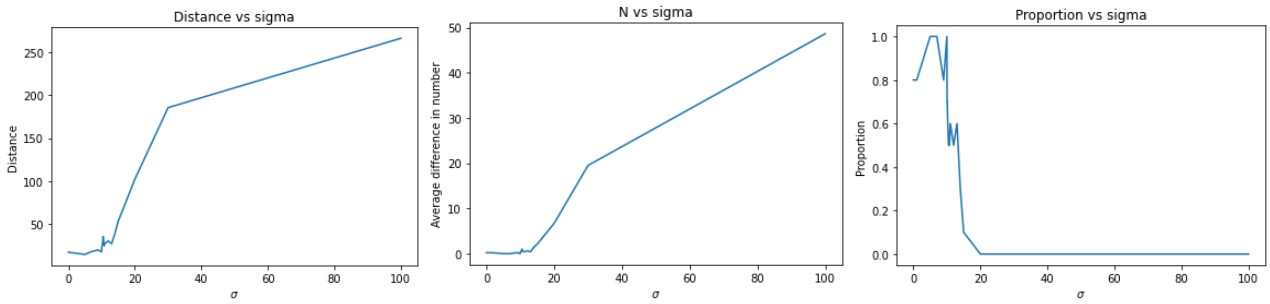
model	Distance between cps	N	$\bar{d}_{\tau-\hat{\tau}}$	$\bar{d}_{\lvert N-\hat{N}\rvert}$	Prop($\hat{N}=N$)
Non-subgaussian data	200-300	60	29.76	3.5	0.2
Subgaussian data	200-300	60	15.95	0.2	0.8

In certain cases, the goal is to estimate change points with higher precision by minimizing the distance between the true and estimated change point locations. For data sets where the output possesses a sufficiently large number of moments l, reducing the test window size T_0 can enhance the accuracy of change point localization. This effect is demonstrated in Table 2.4. The input, output, and model function used in the corresponding experiment are identical to those in the feed-forward network setting described in the first simulation. As shown in Table 2.4, decreasing the window size T_0 results in more precise estimation of the change point locations.

Table 2.4: Precise train

model	Distance between cps	N	$\bar{d}_{\tau-\hat{\tau}}$	T_0
First train	300-1500	10	20.46	50
Precise train	300-1500	10	8.43	20

Finally, the theorem's assumption that the signal must be significantly larger than the standard deviation of the error term σ is essential, as demonstrated in the proof provided in the supplementary section. Figure 2.3 illustrates the impact of σ's magnitude on the algorithm's performance. This relationship is evaluated using three metrics: the average distance between true and estimated change points $\bar{d}_{\tau-\hat{\tau}}$, the average difference between the number of detected and true change points $\bar{d}_{\lvert N-\hat{N}\rvert}$ and the proportion of instances where the exact number of change points is detected Prop($\hat{N}=N$). We observe that as the magnitude of σ increases, the detection algorithm's performance deteriorates significantly, highlighting the critical importance of the signal magnitude assumption in ensuring algorithmic effectiveness.



(a) $\bar{d}_{\tau-\hat{\tau}}$ vs σ (b) $\bar{d}_{|N-\hat{N}|}$ vs σ (c) Prop($\hat{N} = N$) vs σ

Figure 2.3: Performance metrics vs σ

2.5.2 Experiments with Real Data Sets

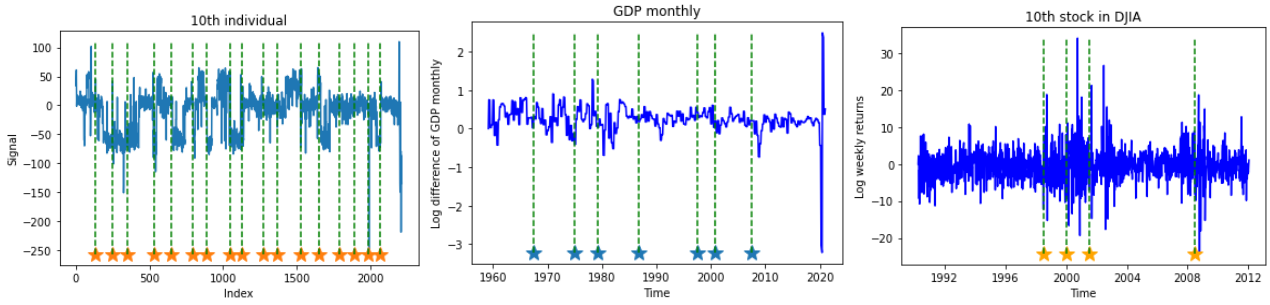
Next, we present the results of applying our detection strategy to a variety of real-world datasets.

The first data set consists of data from 43 individuals with bladder tumors, where each value represents the copy number of specific gene mutations for each segment of each individual. The data set contains 2216 segments [16]. Using the same detection strategy as in the VAR model, we apply a window size of $T_0 = 50 \simeq \sqrt{2216}$. Our two-step algorithm detected 17 change points in this data set. Figure (a) in Figure 2.4 illustrates the number of mutation copies related to bladder cancer, with the detected change points marked by stars and vertical dashed lines.

The second data set contains the log-weekly returns for 29 stocks comprising the Dow Jones Industrial Average (DJIA). The dataset spans 1138 time points, from April 1990 to January 2012, where each time point represents one week [49].For this dataset, we use a window size of $T_0 = 30 \simeq \sqrt{1138}$. Our two-step algorithm successfully detected changes near the years 1998, 2000, 2002, and 2008, corresponding to significant events such as the 1997 financial crisis, the collapse of the technology bubble, the stock market downturn of 2002, and the financial crisis of 2008. Figure (c) in Figure 2.4 shows the log weekly returns for the 10th stock used to calculate the DJIA index, with detected change points highlighted.

The third data set, sourced from [14], consists of 743 time points, where each time point

represents a month. The 1-dimensional output represents the log-difference in monthly GDP, while the 122-dimensional input comprises various economic indicators, including stock market data, real personal income, and US treasury bond returns. This dataset spans several key recessions in U.S. economic history. Our algorithm detected change points corresponding to events such as the 1973 oil crisis, the 1980 recession, the dot-com bubble of 2000, and the great recession of 2008, among others.



(a) The genes mutations of Cancer patients (b) log difference of GDP (c) DJIA Stock market index

Figure 2.4: Real data

2.5.3 Comparison with commonly used algorithms

We compared the proposed detection algorithm with other popular change point algorithms, including KLCPD [20], Sparsified Binary Segmentation [26] and double CUSUM in [25].

The synthetic data used for this comparison are generated according to a VAR(5), a Non-linear VAR [66], and a multiple species Lotka-Volterra system of coupled differential equations. The VAR(5) model is defined similarly to the model in Section 2.5 equation (2.8) and has dimension 200.

The non-linear VAR model described in citeNonlinearVAR is defined as:

$$x_t = Ax_{t-1} + \sum_{i=0}^{q} \Lambda_i \rho_i (f_{rt-i} + \epsilon_t) \tag{2.10}$$

where

$$f_t = \Gamma_1 f_{t-1} + \Gamma_2 f_{t-2} + \epsilon_t^f \tag{2.11}$$

Here Γ_j, ϵ_t^f are the coefficient matrices and the error process for a VAR(2) model f_t and A and ϵ_t are the coefficient matrix and error process for the non-linear VAR model x_t, respectively. Further, Λ_i is the coefficient matrix for the non-linear lag terms in the model and ρ_i is the corresponding non-linear function. In the experiment, we set the number of lags to $q = 6$, the dimension of f_t to 10, and the dimension of x_t to be 200. We also set the frequency parameter to $r = 3$ aiming to skip direct connections between the consecutive components f_t, f_{t-1}.

The data from the Lotka-Volterra model (denoted as LV in 2.5) described in [66] are generated according to the following differential equations:

$$\frac{dx^i}{dt} = \alpha x^i - \beta x^i \sum_{j \in Pa(x^i)} y^j - \alpha (x^i)^2, 1 \leq i \leq p \tag{2.12a}$$

$$\frac{dy^j}{dt} = \delta y^j \sum_{k \in Pa(y^j)} x^k - \gamma (y^j), 1 \leq j \leq p \tag{2.12b}$$

where x^i corresponds to the population sizes of prey species and y^j is the population sizes of predator species, $\alpha, \beta, \delta, \gamma$ are the fixed parameters controlling strengths of interactions, and the $Pa(x^i), Pa(y^j)$ are the sets of Granger-causal variables of x^i, y^j.

The distance metrics used in the evaluation are: (i) the regular distance $d_{\tau - \hat{\tau}}$ used in Section 2.5, (ii) the sum version of the Hausdorff distance

$$d_{H-W} = \max_{\tau_i \in T} \sum_{\hat{\tau}_i \in \hat{T} | \tau_i - \hat{\tau}_j | <= \min_k |\tau_k - \hat{\tau}_j|} |\tau_i - \hat{\tau}_j| \tag{2.13}$$

, and (iii) the product version of the Hausdorff distance

$$d_{H-P} = \max_{\tau_i \in T} \prod_{\hat{\tau}_i \in \hat{T} | \tau_i - \hat{\tau}_j | <= \min_k |\tau_k - \hat{\tau}_j|} |\tau_i - \hat{\tau}_j| \tag{2.14}$$

where $T, \hat{T}$ correspond to the set of true and detected change points, respectively. For the comparison, we also report precision and recall. In addition, we use two modified versions of the Hausdorff distance. While many algorithms achieve strong performance in terms of overall distance and accuracy, they often tend to produce multiple estimates clustered around a true change point. In practice, this behavior is problematic, as it is unclear which estimate should be selected when the true change point is unknown. By introducing these refined metrics, we penalize such behavior, ensuring that an algorithm cannot achieve high performance simply by overestimating around true change points. This allows for a more informative assessment of both the accuracy and robustness of different change point detection algorithms. In the experiments, the distances between successive change points are randomly generated from a uniform distribution over the interval $[1200, 1500]$.

The results of the comparison experiments are presented in Table 2.5. The Table reports five evaluation metrics for each combination of detection method and dataset. Our proposed method outperforms the three competing approaches on two of the datasets. For the Nonlinear VAR dataset, however, the Double CUSUM method exhibits better performance for selective metrics.

2.6 Conclusion and Discussion

The paper introduces a two-step framework for offline change point detection using feed-forward neural networks, combining model training with an error-based detection step. The method comes with theoretical guarantees on the number and locations of detected change points, including bounds on estimation error and exact recovery conditions. Extensive experiments on both synthetic and real-world datasets validated the method's accuracy and robustness. The algorithm performs well in various simulated settings, including feed-forward networks, random forests, and vector autoregressive models. In real-world data—such as genetic copy number variations, DJIA stock returns, and GDP indicators—it successfully

Table 2.5: Comparison of Methods on Various Models

Method	Data	$\bar{d}_{\tau-\hat{\tau}}$	d_{H-W}	d_{H-P}	precision	recall
ML model	Nonlinear VAR	118.16	**134.0**	31.7	0.675	0.70
Sparsified Binary Segmentation	Nonlinear VAR	157.94	400.2	284.5	0.362	**0.975**
Double Cusum	Nonlinear VAR	**90.00**	130.20	**8.9**	**0.704**	0.925
KLCPD	Nonlinear VAR	499.36	302.7	1108.5	0.09	0.05
ML model	LV	**87.84**	**132.90**	**48.2**	**0.58**	**0.75**
Sparsified Binary Segmentation	LV	480.84	391.0	5803.7	0.11	0.65
Double Cusum	LV	561.25	850.0	273.70	0.0	1.0
KLCPD	LV	1601.09	209.4	inf	0.0	0.0
ML model	VAR	**20.05**	**32.5**	**32.5**	**0.90**	**0.875**
Sparsified Binary Segmentation	VAR	480	391.0	5803.7	0.11	0.65
Double Cusum	VAR	561.25	850.0	273.70	0.0	1.0
KLCPD	VAR	862.64	284.0	inf	0.09	0.1

identified meaningful structural shifts. Compared with existing methods like KLCPD and Sparsified Binary Segmentation, the proposed approach showed superior performance on most data sets.

While the framework is adaptable to architectures like RNNs or CNNs, current theoretical results do not extend to these models. Future work could address this, as well as improve precision under temporal dependence by leveraging structures like martingales or Markov chains. Scalability to large datasets and online detection is another promising direction, especially for real-time applications in finance or anomaly detection.

In summary, our framework offers a principled and effective solution for offline change point detection, with strong empirical performance and clear paths for future development.

2.7 Proofs

2.7.1 Proof of Theorem 1

We start by noting that some of the auxiliary lemmas used in the proof and presented in the Supplement for completeness, assume that the output of the neural network to be one dimensional. However, in the detection algorithm is h-dimensional. To mitigate this difference, we consider each feed-forward network employed as $\hat{f} = (\hat{f}_1, ..., \hat{f}_h)$, where each $\hat{f}_k \in \mathcal{F}(L, \boldsymbol{p}, s, F)$ represents a network defined the same way as in Lemma 5 and similarly $f_0 = (f_0^1, ..., f_0^h)$ where$f_0^k \in \mathcal{G}(q, \boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\beta}, K)$. Then,

$$\mathbb{E}_{f_0}[||\hat{f}(X) - f_0(X)||^2] = \sum_{k=1}^{h} \mathbb{E}_{f_0}[(\hat{f}_k(X) - f_0^k(X))^2] \tag{2.15}$$

where each term in the sum is controlled by the resutls in Lemma 5 and the sum will also be controlled with multiplication by h.

For the single change point case in Theorem 1, τ is the estimated change point where $\tau = \arg\max_t(E(t))$. Start by assuming that $T_0 \gg C|\tau - \tau^*|$ for some constant C.

Then, if $\tau^* > \tau$:

$$E(\tau) - E(\tau^*) = \sum_{i=\tau}^{\tau+T_0} ||\hat{f}_\tau^1(X_i) - Y_i||^2 - \sum_{i=\tau^*}^{\tau^*+T_0} ||\hat{f}_{\tau^*}^2(X_i) - Y_i||^2 \geq 0, \tag{2.16}$$

where $\hat{f}_\tau^1 \in arg\min_{f \in \mathcal{F}(L,p,s,F)} \sum_{i \leq \tau}(Y_i - f(X_i))^2$ and $\hat{f}_{\tau^*}^2 \in arg\min_{f \in \mathcal{F}(L,p,s,F)} \sum_{i \leq \tau^*}(Y_i - f(X_i))^2$

Hence,

$$\sum_{i=\tau}^{\tau^*} ||\hat{f}_\tau^1(X_i) - Y_i||^2 + \sum_{i=\tau^*}^{\tau+T_0} ||\hat{f}_\tau^1(X_i) - Y_i||^2 - \sum_{i=\tau^*}^{\tau+T_0} ||\hat{f}_{\tau^*}^2(X_i) - Y_i||^2 - \sum_{i=\tau+T_0}^{\tau^*+T_0} ||\hat{f}_{\tau^*}^2(X_i) - Y_i||^2 \geq 0$$

$$\sum_{i=\tau}^{\tau^*} ||\hat{f}_\tau^1(X_i) - Y_i||^2 + \sum_{i=\tau^*}^{\tau+T_0} (||\hat{f}_\tau^1(X_i) - Y_i||^2 - ||\hat{f}_{\tau^*}^2(X_i) - Y_i||^2) - \sum_{i=\tau+T_0}^{\tau^*+T_0} ||\hat{f}_{\tau^*}^2(X_i) - Y_i||^2 \geq 0 \quad (2.17)$$

Denote by $I_1 = \sum_{i=\tau}^{\tau^*} ||\hat{f}_\tau^1(X_i) - Y_i||^2$, $I_2 = \sum_{i=\tau^*}^{\tau+T_0} (||\hat{f}_\tau^1(X_i) - Y_i||^2 - ||\hat{f}_{\tau^*}^2(X_i) - Y_i||^2)$ and $I_3 = \sum_{i=\tau+T_0}^{\tau^*+T_0} ||\hat{f}_{\tau^*}^2(X_i) - Y_i||^2$

For I_1:

$$I_1 \leq \sum_{i=\tau}^{\tau^*} (||\hat{f}_\tau^1(X_i) - f_0(X_i)||^2 + ||Y_i - f_0(X_i)||^2)$$

$$\leq (\tau^* - \tau)\mathbb{E}_{f_0}[(\hat{f}_\tau^1(X) - f_0(X)^2] + o(\frac{1}{\tau^* - \tau}) + \sum_{\tau}^{\tau^*} ||Y_i - f_0(X_i)||^2$$

$$\overset{Lemma5}{\leq} C_1 \phi_T L \log^2 T(\tau^* - \tau) + h\sigma^2(\tau^* - \tau) \quad (2.18)$$

For I_2:

$$
\begin{aligned}
I_2 &= \sum_{i=\tau^*}^{\tau+T_0} (||\hat{f}_\tau^1(X_i) - Y_i||^2 - ||\hat{f}_{\tau^*}^2(X_i) - Y_i||^2) \\
&\overset{CauchyInequity}{\leq} \sum_{i=\tau^*}^{\tau+T_0} (||\hat{f}_\tau^1(X_i) - \hat{f}_{\tau^*}^2(X_i)||)(||\hat{f}_\tau^1(X_i) + \hat{f}_{\tau^*}^2(X_i) - 2Y_i||) \\
&\overset{CauchyInequity}{\leq} \sqrt{T_0 E_{f_1}[||\hat{f}_\tau^1(X) - \hat{f}_{\tau^*}^2(X)||^2] + o(\frac{1}{T_0})} \\
&\quad \sqrt{\Big(\sum_{i=\tau^*}^{\tau+T_0} ||\hat{f}_\tau^1(X_i) - f_0(X_i)||^2 + ||\hat{f}_{\tau^*}^2(X_i) - f_0(X_i)||^2 + 2||Y_i - f_0(X_i)||^2 \Big)} \\
&\overset{Lemma5}{\leq} \sqrt{C_2 T_0 \phi_T L \log^2 T} \\
&\quad \sqrt{\Big(\sum_{i=\tau^*}^{\tau+T_0} ||\hat{f}_\tau^1(X_i) - f_0(X_i)||^2 + ||\hat{f}_{\tau^*}^2(X_i) - f_0(X_i)||^2 + 2||Y_i - f_0(X_i)||^2 \Big)} \qquad (2.19) \\
&\overset{MinkovskyInequity}{\leq} \sqrt{C_2 T_0 \phi_T L \log^2 T} \\
&\quad (\sqrt{\sum_{i=\tau^*}^{\tau+T_0} ||\hat{f}_\tau^1(X_i) - f_0(X_i)||^2 + ||\hat{f}_{\tau^*}^2(X_i) - f_0(X_i)||^2} \\
&\quad + \sqrt{\sum_{i=\tau^*}^{\tau+T_0} 2||Y_i - f_1(X_i)||^2 + 2||f_0(X_i) - f_1(X_i)||^2}) \\
&\leq \sqrt{C_2 T_0 \phi_T L \log^2 T} (\sqrt{2T_0 \mathbb{E}_{f_1}[||\hat{f}_{\tau^*}^2(X) - f_0(X)||^2] + o(\frac{1}{T_0})} \qquad (2.20) \\
&\quad + \sqrt{\sum_{i=\tau^*}^{\tau+T_0} 2||Y_i - f_1(X_i)||^2 + 2||f_0(X_i) - f_1(X_i)||^2}) \\
&\overset{Lemma5}{\leq} \sqrt{C_2 T_0 \phi_T L \log^2 T} \Big(\sqrt{2C_2 T_0 \phi_T L \log^2 T} + \sqrt{\sum_{i=\tau^*}^{\tau+T_0} 2T_0 h\sigma^2 + 2M_2 T_0 + o(\frac{1}{T_0})} \Big) \\
&\leq 2C_2 T_0 \phi_T L \log^2 T + \sqrt{2C_2 \phi_T L \log^2 T h} T_0 \sigma + \sqrt{2C_2 \phi_T L \log^2 T M_2} T_0 \\
&\leq C_4(M_2, C_2, L) T_0 \phi_T \log^2 T,
\end{aligned}
$$

where C_4 is a constant given M_2, C_2 and L.

For I_3:

$$
\begin{aligned}
I_3 &= \sum_{i=\tau+T_0}^{\tau^*+T_0} ||\hat{f}^2_{\tau^*}(X_i) - Y_i||^2 \\
&\geq \sum_{i=\tau+T_0}^{\tau^*+T_0} (||f_0(X_i) - f_1(X_i)||^2 - ||\hat{f}^2_{\tau^*}(X_i) - f_0(X_i)||^2 - ||f_1(X_i) - Y_i||^2) \\
&\geq (\tau^* - \tau)\mathbb{E}_{f_0}[(f_0(X) - f_1(X))^2] - (\tau^* - \tau)\mathbb{E}_{f_1}[(\hat{f}^2_{\tau^*}(X) - f_1(X))^2] \\
&\quad - (\tau^* - \tau)\mathbb{E}[\epsilon^2] + o(\frac{1}{\tau^* - \tau}) \overset{Lemma5}{\geq} (\tau^* - \tau)(M_1 - h\sigma^2 - C_3\phi_T L \log^2 T) \quad (2.21)
\end{aligned}
$$

By the previous equation $I_1 + I_2 \geq I_3$

$$
\begin{aligned}
C_1\phi_T L \log^2 T(\tau^* - \tau) + h\sigma^2(\tau^* - \tau) + C_4(M_2, C_2, L)T_0^2\phi_T \log^2 T \\
\geq (\tau^* - \tau)(M_1 - h\sigma^2 - C_3\phi_T L \log^2 T) \\
|\tau^* - \tau| \leq \frac{C_4(M_2, C_2, L)T_0\phi_T \log^2 T}{M_1 - 2h\sigma^2 - (C_1 + C_3)\phi_T L \log^2 T} \quad (2.22)
\end{aligned}
$$

Similarly, if $\tau > \tau^*$, we can prove (details provided in A.1.4):

$$
|\tau - \tau^*| \leq \frac{C_1\phi_T L \log^2(T) + T_0(2h\sigma^2 - M_2 + \min_{i=1,\ldots,L} p_i \frac{\log^2(\tau - \tau^*)}{\tau - \tau^*})}{M_1 - 2h\sigma^2 - C_3\phi_T L \log^2 T - o(1)}. \quad (2.23)
$$

Here, if $|\tau - \tau^*| \geq \sqrt{T_0}$, there $\exists\ C_5$ constant, such that $T_0 \min_{i=1,\ldots,L} p_i \frac{\log^2(\tau - \tau^*)}{\tau - \tau^*} \leq C_5 T_0$

$$
|\tau - \tau^*| \leq \frac{2T_0h\sigma^2 - T_0M_2 + C_5T_0 + C_1\phi_T L \log^2(T)}{M_1 - 2h\sigma^2 - C_3\phi_T L \log^2 T} \leq \frac{C_5T_0 + C_1\phi_T L \log^2(T)}{M_1 - 2h\sigma^2 - C_3\phi_T L \log^2 T} \quad (2.24)
$$

Next, if $|\tau - \tau^*| > CT_0$, and if $\tau^* > \tau$.

$$
\begin{aligned}
E(\tau) - E(\tau^*) &= \sum_{i=\tau}^{\tau+T_0} ||\hat{f}^1_{\tau}(X_i) - Y_i||^2 - \sum_{i=\tau^*}^{\tau^*+T_0} ||\hat{f}^2_{\tau^*}(X_i) - Y_i||^2 \\
&\overset{Lemma5}{\leq} 2T_0C\phi_T L \log^2 T + 2T_0h\sigma^2 - M_1T_0 < 0 \quad (2.25)
\end{aligned}
$$

If on the other hand $\tau > \tau^*$

$$E(\tau) - E(\tau^*) = \sum_{i=\tau}^{\tau+T_0} ||\hat{f}_\tau^1(X_i) - Y_i||^2 - \sum_{i=\tau^*}^{\tau^*+T_0} ||\hat{f}_{\tau^*}^2(X_i) - Y_i||^2$$
$$\overset{Lemma5}{\leq} 2T_0 C\phi_{T_0} L\log^2(T_0) + 2T_0 h\sigma^2 - M_1 T_0 < 0 \quad (2.26)$$

Hence, a contradiction that $E(\tau) \geq E(\tau^*)$.

In summary, there exists constant C_5, C, with probability $1 - \frac{C}{T_0}$

$$|\tau^* - \tau| \leq \frac{C_5 T_0}{M_1 - 2h\sigma^2} \quad (2.27)$$

According to the proof in section 2.7.1, the probability

$$P\bigg(|\sum_{i=\tau}^{\tau+T_0} ||\hat{f}_\tau^1(X_i) - f_1(X_i)||^2 - T_0\mathbb{E}[||\hat{f}_\tau^1(X_i) - f_1(X_i)||^2]| > \delta\bigg) \leq \frac{C(\delta)}{T_0} \quad (2.28)$$

given $\forall\delta$, C is a constant depending on δ. For other approximations in the proof, the technique is similar. This probability will be further specified in the proof of multiple change point cases in the next Section, given the different assumptions of the model and data.

2.7.2 Proof of theorem 2

Let T_0 be the test window size. $\hat{\tau}_i$ denotes the i^{th} estimate change point. By the proof of a single change point,

$$|\hat{\tau}_i - \tau_i| \leq \max(\frac{C_{f_i}(M2_i)T_0\phi_{|\tau_{i+1}-\tau_i|}\log^2|\tau_{i+1}-\tau_i|}{M1_i - 2h\sigma^2 - C\phi_{|\tau_{i+1}-\tau_i|}L\log^2|\tau_{i+1}-\tau_i|},$$
$$\frac{C_5T_0 + C_1\phi_{|\tau_{i+1}-\tau_i|}L\log^2(|\tau_{i+1}-\tau_i|)}{M1_i - 2h\sigma^2 - C_3\phi_{|\tau_{i+1}-\tau_i|}L\log^2|\tau_{i+1}-\tau_i|}, \sqrt{T_0})$$
$$\leq \max\left(\frac{C_{f_i}(M2_i)T_0\phi_T\log^2 T}{M1_i - 2h\sigma^2}, \frac{C_5T_0 + C_1\phi_T L\log^2(T)}{M1_i - 2h\sigma^2}, \sqrt{T_0}\right) \leq C\frac{C_5}{M1_i - 2h\sigma^2}T_0 \quad (2.29)$$

By Lemma 9, with probability $1 - \frac{C}{T_0^{p-1}}$ if the data X are independent, and $1 - \frac{C}{T_0}$ for dependent X, the above inequity will hold.

For X independently distributed and sub-Gaussian, by Lemma 7, the probability will be $1 - Ce^{-C_1 T_0}$.

Inside the algorithm 2, define $T_0 = \sqrt[\rho]{T} = o(T)$, and for each peak found, define the peak window as $C_6 T_0 = C\frac{C_5}{M1_* - 2h\sigma^2} T_0$ and $\phi = \frac{M1_*}{2} - 2h\sigma^2$. Then, inside the interval $[\min(t - C_6 T_0, 0), (t + C_6 T_0)]$ when detecting peak at time t, there is one and only one change point with high probability depending on the type of input and model because windows of the nearby two peaks detected $[\min(t' - C_6 T_0, 0), (t' + C_6 T_0)]$ which containing the relative true change points will not intersect with it. Also, the next detection window will be at least one detection window size $C_6 T_0$ away from the previous window, and will not detect the previous change point for the second time.

Next, we must also show that near each change point, the algorithm identifies a peak in the error function. Given any change point $\tau_i, 1 \leq i \leq N$.
Let$\hat{f}_{\tau_i - 2T_0} \in arg\min_{f \in \mathcal{F}(L,p,s,F)} \sum_{k=\tau_{i-1}}^{\tau_i - 2T_0} ||Y_k - f(X_k)||^2$,
$\hat{f}_{\tau_i - T_0} \in arg\min_{f \in \mathcal{F}(L,p,s,F)} \sum_{k=\tau_{i-1}}^{\tau_i - T_0} ||Y_k - f(X_k)||^2$,
$\hat{f}_{\tau_i} \in arg\min_{f \in \mathcal{F}(L,p,s,F)} \sum_{k=\tau_{i-1}}^{\tau_i} ||Y_k - f(X_k)||^2$,
$\hat{f}_{\tau_i + T_0} \in arg\min_{f \in \mathcal{F}(L,p,s,F)} \sum_{k=\tau_i}^{\tau_i + T_0} ||Y_k - f(X_k)||^2$ and
$\hat{f}_{\tau_i + 2T_0} \in arg\min_{f \in \mathcal{F}(L,p,s,F)} \sum_{k=\tau_i}^{\tau_i + 2T_0} ||Y_k - f(X_k)||^2$. To detect a peak near the true change point, the following result holds:

$$\min(E(\tau_i) - E(\tau_i - T_0), E(\tau_i) - E(\tau_i + T_0)) \gg \max\Big(|E(\tau_i - T_0) - E(\tau_i - 2T_0)|, |E(\tau_i + T_0) - E(\tau_i + 2T_0)|\Big) \quad (2.30)$$

which implies

$$\begin{aligned}
&\min(\sum_{k=\tau_i}^{\tau_i+T_0}||Y_k-\hat{f}_{\tau_i}(X_k)||^2-\sum_{k=\tau_i-T_0}^{\tau_i}||Y_k-\hat{f}_{\tau_i-T_0}(X_k)||^2,\\
&\sum_{k=\tau_i}^{\tau_i+T_0}||Y_k-\hat{f}_{\tau_i}(X_k)||^2-\sum_{k=\tau_i+T_0}^{\tau_i+2T_0}||Y_k-\hat{f}_{\tau_i+T_0}(X_k)||^2)\\
\gg&\max(|\sum_{k=\tau_i-T_0}^{\tau_i}||Y_k-\hat{f}_{\tau_i-T_0}(X_k)||^2-\sum_{k=\tau_i-2T_0}^{\tau_i-T_0}||Y_k-\hat{f}_{\tau_i-2T_0}(X_k)||^2|\\
&,|\sum_{k=\tau_i+T_0}^{\tau_i+2T_0}||Y_k-\hat{f}_{\tau_i+T_0}(X_k)||^2-\sum_{k=\tau_i+2T_0}^{\tau_i+3T_0}||Y_k-\hat{f}_{\tau_i+2T_0}(X_k)||^2|)
\end{aligned}\tag{2.31}$$

Let $J_1=\sum_{k=\tau_i}^{\tau_i+T_0}||Y_k-\hat{f}_{\tau_i}(X_k)||^2-\sum_{k=\tau_i-T_0}^{\tau_i}||Y_k-\hat{f}_{\tau_i-T_0}(X_k)||^2$,
$J_2=\sum_{k=\tau_i}^{\tau_i+T_0}||Y_k-\hat{f}_{\tau_i}(X_k)||^2-\sum_{k=\tau_i+T_0}^{\tau_i+2T_0}||Y_k-\hat{f}_{\tau_i+T_0}(X_k)||^2$,
$J_3=|\sum_{k=\tau_i-T_0}^{\tau_i}||Y_k-\hat{f}_{\tau_i-T_0}(X_k)||^2-\sum_{k=\tau_i-2T_0}^{\tau_i-T_0}||Y_k-\hat{f}_{\tau_i-2T_0}(X_k)||^2|$ and
$J_4=|\sum_{k=\tau_i+T_0}^{\tau_i+2T_0}||Y_k-\hat{f}_{\tau_i+T_0}(X_k)||^2-\sum_{k=\tau_i+2T_0}^{\tau_i+3T_0}||Y_k-\hat{f}_{\tau_i+2T_0}(X_k)||^2|$.

For J_1:

$$\begin{aligned}
J_1=&\sum_{k=\tau_i}^{\tau_i+T_0}||Y_k-\hat{f}_{\tau_i}(X_k)||^2-\sum_{k=\tau_i-T_0}^{\tau_i}||Y_k-\hat{f}_{\tau_i-T_0}(X_k)||^2\\
\geq&\sum_{k=\tau_i}^{\tau_i+T_0}(||f_{i+1}(X_k)-f_i(X_k)||^2-||Y_k-f_{i+1}(X_k)||^2)-\sum_{k=\tau_i}^{\tau_i+T_0}||f_i(X_k)-\hat{f}_{\tau_i}(X_k)||^2\\
&-\sum_{k=\tau_i-T_0}^{\tau_i}(||Y_k-f_i(X_k)||^2+||f_i(X_k)-\hat{f}_{\tau_i-T_0}||^2)\\
\overset{Theorem5}{\geq}&o(1)+M1_iT_0-C_1T_0\phi_{\tau_i-\tau_{i-1}}L\log^2(\tau_i-\tau_{i-1})\\
&-C_2T_0\phi_{\tau_i-\tau_{i-1}-T_0}L\log^2(\tau_i-\tau_{i-1}-T_0)-2T_0h\sigma^2
\end{aligned}\tag{2.32}$$

The proof for the terms J_2, J_3, J_4 are similar and is provided in the Supplement (see Section A.1.4).

By the previous equation, $T_0 \ll T \leq \tau_i-\tau_{i-1}$, $\min(J_1,J_2)=J_2, \max(J_3,J_4)=J_4$. To let

$\min(J_1, J_2) \gg \max(J_3, J_4)$ which is $J_2 \gg J_4$.

$$o(1) + M1_i T_0 - 2T_0 h\sigma^2 - C_1 T_0 \phi_{\tau_i - \tau_{i-1}} L \log^2(\tau_i - \tau_{i-1}) - C_2 T_0 \phi_{T_0} L \log^2(T_0)$$
$$\gg o(1) + 2T_0 h\sigma^2 + C_1 T_0 \phi_{T_0} L \log^2 T_0 + C_2 T_0 \phi_{2T_0} L \log^2(2T_0) \quad (2.33)$$

To satisfy the condition above,

$$T_0 \gg \frac{o(1)}{M1_i - 4h\sigma^2 - C\phi_{T_0} L \log^2(T_0)} \quad (2.34)$$

This will be satisfied as $T_0 \to \infty$ when $T_{\text{sum}} \to \infty$ and then the algorithm will detect a peak near the true change point with probability $1 - \frac{C}{T_0}$ (or $1 - \frac{C}{T_0^{p-1}}$ and $1 - e^{-CT_0}$) for some constant C depending on the type of input and model. And when $\phi = \frac{M1_*}{2} - 2h\sigma^2$, $J_2 - J_4 > \phi$ thus finishing this part of proof.

Next, we proceed to compute the probability of accurate approximation. Based on the given assumptions, we can complete the proof by considering the cases separately.

For independent variables (*Case I*), each change point is detected by the algorithm with probability at least $1 - \frac{C}{T_0^{p-1}}$. Consequently, with probability at least $(1 - \frac{C}{T_0^{p-1}})^N$, the algorithm will detect all N change points —and only those N—with the distance between each estimated and true change point being properly controlled.

$$(1 - \frac{C}{T_0^{p-1}})^N \geq 1 - \frac{CN}{T_0^{p-1}} \geq 1 - \frac{CT_{\text{sum}}}{TT_0^{p-1}} \geq 1 - \frac{C_0 T_{\text{sum}}}{T^{1+\rho(p-1)}} \geq 1 - C_0 T_{\text{sum}}^{-\frac{1}{\rho(p-1)}} \underset{T_{\text{sum}} \to \infty}{\to} 1 \quad (2.35)$$

The proof for Case II and Case III is similar and included for completeness in the Supplementary Section A.1.4. This completes the proof of the Theorem.

Chapter 3

Graph Neural Network-Based Nonparametric Temporal-Spatial Graph Model

3.1 Introduction

In this chapter, we will discuss and propose a nonparametric temporal-spatial neural network-based graph network model that can handle both existing node points and emerging new node points. In many real-world applications, one often seeks to model a large collection of time series that are interconnected through an underlying *network structure*. Motivated by problems arising in social network analysis, [104] has introduced the network vector autoregression (NAR) model, which captures temporal dependencies in each node's time series not only through its own past values but also through those of its neighboring nodes. Building upon this framework, [99] has proposed a general formulation that encompasses various extensions of the basic NAR model for numerical time series, while [10] adapted the NAR framework to accommodate count data by developing a Poisson-based version of the model.

However, these models assume that the temporal dynamics are *linear* or captured by a simple *parametric* statistical model. There have been selected approaches to extend the above framework to capture nonlinearities. For example, threshold vector autoregressive (TVAR) models [91] generalize their univariate counterparts [89] to the multivariate setting, allowing different linear dynamics to govern the evolution of a vector time series depending on the regime determined by an observed threshold variable. These models are particularly well-suited for capturing *regime-switching* behavior and their statistical properties and estimation procedures have been extensively studied in works such as [91] and [61]. Even though such models exhibit richer temporal dynamics, they are still linear within time segments. Beyond the basic TVAR model, various generalizations have been proposed in the statistical literature to account for richer nonlinearities and dynamic structures. The multiple regime TVAR models allow for more than two regimes separated by multiple thresholds, extending the binary switching nature of the original TVAR model [37]. The smooth transition vector autoregressive (STVAR) models, such as those proposed by [84] and further developed in [64], relax the abrupt regime-switching assumption by allowing transition between regimes to occur smoothly based on a continuous transition variable. Another notable extension includes Markov-switching vector autoregressive (MS-VAR) models [39], where regime changes are governed by a latent Markov process, providing a probabilistic framework for modeling regime transitions. Additionally, models such as the SETAR-VAR (self-exciting threshold VAR) and TVAR with exogenous thresholds (TVARX) introduce additional flexibility by incorporating lagged endogenous or exogenous variables as thresholding mechanisms [21, 61].

In another stream of literature, the integration of deep learning methodologies with network-based approaches has revolutionized temporal dynamics modeling in recent years. Graph Neural Networks (GNNs) represent a powerful class of neural network architectures specifically designed to process data structured as graphs [97]. Their application to time series analysis has emerged as a promising frontier, offering novel solutions to capture complex temporal dependencies, spatial correlations, and nonlinear dynamics simultaneously [98].

Temporal Graph Networks (TGNs) [63, 77], provide a framework for deep learning on dynamic graphs by representing them as sequences of time-stamped events. TGNs incorporate memory modules and message-passing mechanisms to capture the temporal evolution of node and edge features. Spatio-Temporal Graph Convolutional Networks (ST-GCNs) combine graph convolutions with temporal convolutions to model spatial dependencies between time series and their temporal evolution [100]. Graph Attention Networks (GATs) have been adapted for time series by incorporating attention mechanisms that dynamically weight the importance of different nodes and time steps [92, 102]. Finally, the integration of Neural ODEs with GNNs has led to continuous-time models capable of handling irregularly sampled time series data [72].

In this paper, we will propose a novel method leveraging both the spatial and temporal information. Similar to the ST-GCNs and GMAN [102], our method uses the summation of the temporal embeddings and spatial embeddings as the embeddings in the formulation. However our method updates the node output at each time point t and does not convolve along the entire time axis, and also the summation of temporal and spatial embeddings happens at the final layer, not at each layer. And unlike the TGNs where the update are formed in events at each time point t and passed via message function to nodes and edges, our methods assume each node are updated at every time point t with both node feature and previous node output information from their neighbors and themselves. And unlike the GDEs, our method will focus on the discrete temporal mechanism. Finally, the key components that differentiate our proposed method with these methods are the capability to update not only the existing but also new unknown node points at each time point t. Moreover, our method uses a mixture of GNN and feed-forword neural networks where each network component in our network is composed by multiple layers and each layer is formed by multiple feed-forward layers followed by a GNN layer. In this paper, we will provide the theoretical asymptotic analysis of the algorithm we developed as well.

3.2 Problem formulation

Notation: For any given matrix $X \in \mathbb{R}^{m\times n}$ and $X_{i,j}$ indexed by i and j, $X_{i,\cdot}$ represents the vector composed of all the value in ith row. For a given matrix A, A[i:j,.] represents a submatrix including rows from the ith row to the jth row inclusively. For any $x \in \mathbb{R}$, $\lfloor x \rfloor$ represents the largest integer less than or equal to it. For any matrix or vector X, $||X||_0$ represents the 0 norm, which is the number of nonzero elements, $||X||_\infty$ represents the infinity norm, which is the maximum absolute value in all elements. For function $x \to f(x)$,$|||f|_\infty||_\infty = \sup_{||x||_\infty=1} |f(x)|_\infty$. For any multidimension function with r variables and d dimension $f : \mathbb{R}^r \to \mathbb{R}^d$, we use annotation $\sigma_v : \mathbb{R}^r \to \mathbb{R}^r$ as the Relu activation function with the bias vector $v = \{v_i\}_{i=1}^r \in \mathbb{R}^r$. Given r dimension vector $y = \{y_i\}_{i=1}^r \in \mathbb{R}^r$,

$$\sigma_v y = \sigma_v(y) = \begin{pmatrix} \max(y_1 - v_1, 0) \\ \max(y_2 - v_2, 0) \\ \cdots \\ \max(y_r - v_r, 0) \end{pmatrix}$$

Consider a time evolving network/graph (V_t, E_t), with node set $V_t \subset \mathbb{Z}_+$ and edge set $E_t \subseteq V_t \times V_t$. The connectivity between nodes is a captured by a *weighted adjacency* matrix $A_t = \{a_{ij}, i, j \in V_t\}$, with $a_{ij} = 0$, if edge $(i, j) \notin E_t$. The term time evolving means we assume at each time t, there are n_t (which can be 0) new points denoted as $I_t = \{i\}_{i=m_t+1,m_t+2,\ldots m_t+n_t}$ where $m_t = |V_{t-1}|, m_1 = 0, V_0 = \emptyset, V_{t+1} = V_t \cup I_{t+1}$. In other words, the data of points in I_t are only available after time t. Let $N(i) = \{j \in V_t : (i, j) \in E_t\}$ denote the *neighborhood of node i (all nodes connected to i* and $\mathcal{N}(i) = N(i) \cup \{i\}$ the *extended* neighborhood that also includes node i. Further, with every node $i \in V_t$, a feature vector $z_i \in \mathbb{R}^d$ and time evolving response $X_{t,i} \in \mathbb{R}$ are associated with. We define $\mathbf{X}_t = \{X_{t,i}\}_{i\le m_t+n_t}, X_{t,i} \in \mathbb{R}, t \le T$ is the node response of all available points at time t and $\mathbf{z} = \{z_i\}_{i\le m_T}, z_i \in R^{d_{feature}}$ is the $d_{feature}$ dimensional node feature and $z_{1:m_t}$ represents the available node features at time t. T is the total number of time points, m_t, n_t is the existing and new points at each time t. We assume new points are added to the existed node points for each time t while $X_{t,1:m_t} = \{X_{t,i}\}_{i\in V_{t-1}}$

represents the node response of existing points (points whose previous data is available) at time t and $X_{t,m_t+1:m_t+n_t} = \{X_{t,i}\}_{i\in I_t}$ represents the node response of new points which means their data is not available in the past. Here we have $m_t + n_t = m_{t+1}$, which denotes that the new points will become the existing points for the next time point $t + 1$.

Here we define a new function space $\mathcal{G}(q, \boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\beta}, K, E)$. For function $g \in \mathcal{G}(q, \boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\beta}, K, E)$, we have $g : \mathbb{R}^{n\times d_{in}} \to \mathbb{R}^{n\times d_{out}}$ mapping n node input with dimension d_{in} to n node output with dimension d_{out} and $g = A_E k_q \circ A_E k_{q-1} ... \circ A_E k_0$ to be a composition of β Hölder smooth functions where $q+1 \in \mathbb{Z}_+$ represents number of layers, each function k_j is a multidimensional β-Hölder smooth function(a higher order version of Lipchitz function) and $\boldsymbol{d} = \{d_i\}_{i=1}^{q+1} \in \mathbb{R}^{q+1}$ and $\boldsymbol{t} = \{t_i\}_{i=1}^{q+1} \in \mathbb{R}^{q+1}$ control the dimension and domain of input and output space of each function $k_i : \mathbb{R}^{n\times d_i} \to \mathbb{R}^{n\times d_{i+1}}, d_q = d_{out}, d_0 = d_{in}$ where n is the number of nodes and $S_i \subseteq \mathbb{R}^{n\times t_i} \subseteq \mathbb{R}^{n\times d_i}, t_i \le d_i$ is the domain of this function, d_i is the input dimension and d_{i+1} is the output dimension of each layer. $A_E = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ represents the graph structure where $\tilde{A} = A + I$ is the adjacency matrix with weights and additional self-loop and $\tilde{D}$ is the degree matrix of $\tilde{A}$. Then the A_E matrix is the normalized adjacency matrix with weights. Then for each layer, the neighborhood information is passed to the center. For example, define value $\mathbf{Y} = A_E k_{j-1} \circ ... A_E k_0$ at layer j and $\mathbf{Y} = \{Y_i\}_{i=1}^{d_j}$ for each node i, then the ith element of $A_E k_j(\mathbf{Y})$ is $(A_E k_j(\mathbf{Y}))_i = \sum_{k\in N(i)} w_{ki} k_j(Y_k)$ where w_{ki} is the normalized adjacency weight on edge between node i and node k and passed information of neighborhood of node i to node i.

The β-Hölder smoothness [80] in one dimension function is defined as functions whose partial derivatives exist up to $\lfloor\beta\rfloor$ order and are bounded and the partial derivatives of order $\lfloor\beta\rfloor$ are $\beta - \lfloor\beta\rfloor$ Hölder. Then define the ball of β-Hölder smoothness functions with radius K where K controled both the Hölder radius and domain S_i of each function k_i which is actually

a t_i dimensional ball with radius K:

$$\mathcal{C}_r^{\beta}(D,K) = \{f : D \subset \mathbb{R}^r \to \mathbb{R}; \sum_{\boldsymbol{\alpha}:|\boldsymbol{\alpha}|<\beta} ||\partial^{\boldsymbol{\alpha}} f||_\infty + \sum_{\boldsymbol{\alpha}:|\boldsymbol{\alpha}|=\lfloor\beta\rfloor} \sup_{x,y\in D;x\neq y} \frac{|\partial^{\boldsymbol{\alpha}} f(x) - \partial^{\boldsymbol{\alpha}} f(y)|}{|x-y|_\infty^{\beta-\lfloor\beta\rfloor}} \leq K\} \tag{3.1}$$

where $\partial^{\boldsymbol{\alpha}} = \partial^{\alpha_1}...\partial^{\alpha_r}$ with $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_r) \in \mathbb{N}^r$ where ∂^{α_i} denotes the α_i partial derivative on the i^{th} dimension and $|\boldsymbol{\alpha}| := |\boldsymbol{\alpha}|_1$. We define the function space $\mathcal{G}(q, \boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\beta}, K, E)$ to be:

$$\begin{aligned}\mathcal{G}(q, \boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\beta}, K, E) := \{&g = A_E k_q \circ A_E k_{q-1} \circ ... \circ A_E k_1 \circ A_E k_0 : \\ &k_i = (k_{ij})_j : [a_i, b_i]^{d_i} \to [a_{i+1}, b_{i+1}]^{d_{i+1}}, k_{ij} \in \mathcal{C}_{t_i}^{\beta_i}([a_i, b_i]^{t_i}, K) \\ &\text{where } |a_i| \leq K, |b_i| \leq K, t_i \leq d_i, A_E = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}\}\end{aligned} \tag{3.2}$$

where $\boldsymbol{d} := (d_0, ..., d_{q+1}), \boldsymbol{t} := (t_0, ..., t_q), \boldsymbol{\beta} := (\beta_0, ..., \beta_q)$, $\tilde{A}, \tilde{D}^{-\frac{1}{2}}$ is the normalized adjacency matrix and degree matrix according to weighted edge E respectively.

The details are in section 3.6.3

Consider a time-evolving model $G = (g_1, g_2, g_3)$ defined on this graph structure where both existing points and emerging points depend on both features and responses of their neighborhood nodes. We start from the existing points which will get the feature updated for each time t. The update equation is defined as

$$X_{t,i} = G(\mathbf{X}_{t-1}, z_{1:m_t})_i + \epsilon_{t,i} = g_3(X_{t-1,1:m_t})_i + \epsilon_{t,i}, i = 1, 2 \ldots, m_t, Var(\epsilon_{t,i}) = \sigma^2$$

for existing points (3.3)

where m_t is the number of existing points at each time t,

$g_3 : \mathbb{R}^{m_t \times 1} \to \mathbb{R}^{m_t \times 1} \in \mathcal{G}(q, \boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\beta}, K, E)$, and σ is the standard deviation of the error term. By the definition $g_3 = A_E^{m_t} k_1^3 \circ A_E^{m_t} k_2^3 ... \circ A_E^{m_t} k_q^3$ where $A_E^{m_t}$ is the adjacency matrix with weights in points set $\{1, 2, ..., m_t\}$ and k_i^3 is the β-Hölder smooth function. When t changes, the set of k_i^3 function will not change while the normalized adjacency matrix $A_E^{m_t}$ will update

due to incoming new node points. As a result, for simplicity, we use g_3 to represent all the transformation functions updating the existing points for each time t.

Note that here we use the edge E as one of the parameters in the function space $\mathcal{G}(q, \boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\beta}, K, E)$, but here actually the edges keep updating at each time t and we use E_t represent the real time evolving edges set. On one hand, at each time t, the data of n_t more new points is available and normalized adjacency matrix $A_E^{m_t} = \tilde{D}_t^{-\frac{1}{2}}(\tilde{A}_t)\tilde{D}_t^{-\frac{1}{2}}$ will get updated when the $\tilde{A}_t, \tilde{D}_t$ get updated. However, the structure of function space remained the same and β-Hölder function k_i^3 remained the same as well. So we use E represent the edges for simplicity. On the other hand, this also means for nodes in I_t, the data will only be available starting from time t. In another word, our model is capable of capturing incomplete data.

We can use the neighborhood information and the previous response for existing points. However, we don't have that for new emerging points. If we want to make our model capable of predicting response of new node points, we need to include that update mechanism as well. Consider each time t, we will have n_t new points we've never seen before. To predict the node response of new points, we need to leverage both feature information and node response of the neighborhood nodes. Given the connection information of the new points, the node response of the new points generated at time t is defined by the equation

$$X_{t,i} = G(\mathbf{X}_{t-1}, z_{1:m_t})_i + \epsilon_{t,i} = g_1(z_{1:m_t})_i + g_2(X_{t-1,1:m_t})_i + \epsilon_{t,i}, i = m_t + 1, \ldots, m_t + n_t,$$
$$Var(\epsilon_{t,i}) = \sigma^2 \text{ for new points} \quad (3.4)$$

For new point cases, the function g_1, g_2 are slightly different from existing point cases. Here $g_1 = e_1 \circ v_1 \circ u_1, u_1 : \mathbb{R}^{m_t \times d_{feature}} \to \mathbb{R}^{m_t \times d_{feature}}, u_1 \in \mathcal{G}(q, \boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\beta}, K, E)$ and nonlinear function $v_1 : \mathbb{R}^{m_t \times d_{feature}} \to \mathbb{R}^{m_t \times 1}, e_1 \circ v_1 := A_E[m_t + 1 : m_t + n_t, .]v_1$ where $A_E[i : j, .]$ represents the ith to jth rows of normalized adjacency matrix inclusively. For new points $g_2 = e_1 \circ v_2 \circ g_3$ and $v_2 : \mathbb{R} \to \mathbb{R}$ where the g_3 is shared between existing and new points

and the new points response is the summition of the weighted sum of predicted neighbor existing points and features. Define $G(\mathbf{X}_{t-1}, z) = g_1(z_{1:m_t}) + g_2(\mathbf{X}_{t-1})$ for new points and $G(\mathbf{X}_{t-1}, z) = g_3(X_{t-1,1:m_t})$ for existing points and $X_{t,i} = G(\mathbf{X}_{t-1}, z)_i + \epsilon_{t,i}$.

3.3 Algorithm

The algorithm has the capability to predict both existing and emerging points. For new emerging points, we need both their adjacency information with existing points and their feature information.

Before we dive into the details of the algorithm, we will introduce a new network function space $\mathcal{H}(L, \boldsymbol{p}, s, F, E, r)$ where each function is a mix of graph convolution network and fully connected network with relu function. For each $h \in \mathcal{H}(L, \boldsymbol{p}, s, F, E, r)$, it is composition of r components and each of component contains one GCN layer and several feed forward layers. Specifically, we have $h = l_r \circ l_{r-1} ... \circ l_1$ and each component $l_i(x) = A_E W_{L_{l_i}} \sigma_{v_{L_{l_i}}} W_{L_{l_i}-1} \sigma_{v_{L_{l_i}-1}} ... W_{L_{l_{i-1}+1}} x, i = 1, ...r; L_{l_0} = 1, L_{l_r} = L$ where $A_E W_{L_{l_i}}$ represents the graph convolution network layer and remaining represents fully connected network layers. The number of weight matrices is L which also represents the total number of GCN layers and feed-forward layers combined. The parameter $\boldsymbol{p} = \{p_i\}_{i=1}^{L}$ represents the dimension of each weight matrix. For weight matrix $W_i \in \mathbb{R}^{p_i \times p_{i+1}}, i = 1, ..., L$ and bias vector $v_i \in \mathbb{R}^{p_i}, i = 1, ..., L$, we have $\sum_{j=0}^{L} ||W_j||_0 + |v_j|_0 \leq s$ to control the sparsity. For the entire function h, we have $|||h|_\infty||_\infty \leq F$ to control the L_∞ norm. If we set $A_E = I$, then the function space degenerates to fully connected network space $\mathcal{H}(L, \boldsymbol{p}, s, F, r)$. The details again will be presented in section 3.6.3.

We define a graph network model $H = (h_1, h_2, h_3) = H_\psi$ where $\psi = (\psi_1, \psi_2, \psi_3)$ are the trainable parameters. For existing nodes, we define network $h_3 = h_{\psi_3} \in \mathcal{H}(L, \boldsymbol{p}, s, F, E, r)$ with trainable parameters ψ_3 where $h_{\psi_3} : \mathbb{R}^{m_t \times 1} \to \mathbb{R}^{m_t \times 1}$

For new points, we define $h_1 = A_E[m_t + 1 : m_t + n_t, .]h_{\psi_1}, h_2 = A_E[m_t + 1 : m_t +$

Algorithm 3 Node Prediction

System Initialization
Input node result $X_{t,i}$, node features z_i, Edge E_t, nodes $V_t = \{i\}_{i=1,2,...,m_t+n_t}$,New points set $I_t = \{m_t + 1, m_t + 2..., m_t + n_t\}$,total time T, number of epochs Epoch.
for e=1,2...,Epoch **do**
  for t=1,2,...,T **do**
    $\hat{X}_{t,1:m_t} = h_3(\mathbf{X}_{t-1})$
    $\hat{X}_{t,m_t+1:m_t+n_t} = h_1(z_{1:m_t}) + h_2(\hat{X}_{t,1:m_t})$
  end for
  Loss=$\mathcal{L}_{\psi_1,\psi_2,\psi_3} = \sum_t \sum_{i\in I(t)} ||\hat{X}_{t,i} - X_{t,i}||$
  while ψ_3 not converge **do**
    $\psi_3 \leftarrow ADAM(\mathcal{L}_{\phi_3})$
  end while
  if $I_t \neq \emptyset$ **then**
    while ψ_1, ψ_2 not converge **do**
      $\psi_1 \leftarrow ADAM(\mathcal{L}_{\psi_1})$
      $\psi_2 \leftarrow ADAM(\mathcal{L}_{\psi_2})$
    end while
  end if
end for
output h_1, h_2, h_3

$n_t, .]h_{\psi_2} \circ h_{\psi_3}$ with trainable parameters ψ_1, ψ_2 and shared trainable parameters ψ_3. $h_{\psi_3}, h_{\psi_1} \in \mathcal{H}(L, \boldsymbol{p}, s, F, E, r)$ are graph network and $h_{\psi_2} \in \mathcal{H}(L, \boldsymbol{p}, s, F, r)$ are feed forward network where $h_{\psi_1} : \mathbb{R}^{m_t \times d_{feature}} \to \mathbb{R}^{m_t \times d_{feature}}$ and $h_{\psi_2} : \mathbb{R}^{m_t \times 1} \to \mathbb{R}^{m_t \times 1}$

Inside the existing points algorithm, the existing points is updated for each new time point by function h_3. We can just update only these points when there is no emerging points as well. And this pretrained h_3 function can also be reused in the new points algorithm. And here we define predictions $\hat{X}_{t,i} = H(\mathbf{X}_{t-1}, z_{1:m_t}) = \{H_\psi(X_{t-1,1:m_t}, z_{1:m_t})_i\}_{i\leq m_t+n_t}, t = 1, 2, ..., T$ and $H_\psi(X_{t-1,1:m_t}, z_{1:m_t})_{t,1:m_t} = h_3(X_{t-1,1:m_t})$ for existing points, $H_\psi(X_{t-1,1:m_t}, z_{1:m_t})_{t,(m_t+1):(m_t+n_t)} = h_1(z_{1:m_t}) + h_2(X_{t-1,1:m_t})$ for new points. As a result, $H(\mathbf{X}_{t-1}, z_{1:m_t})_i = h_3(X_{t-1,1:m_t})_i$ for existing points and $H(\mathbf{X}_{t-1}, z_{1:m_t})_i = h_1(z_{1:m_t})_i + h_2(X_{t-1,1:m_t})_i$ for new points.

3.4 Theorem

Given the time-evolving mechanism, we can use the graph neural network to approximate it. And the error of the approximation is bounded when certain assumptions are satisfied.

We will start from the general case that for any function from function space $\mathcal{G}$, we have a mixture network function from function space $\mathcal{H}$ to approximate it.

Theorem 3. *Given input dimension d_{in} and output dimension d_{out}, let $g \in \mathcal{G}(q, \boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\beta}, K, E)$ and $g : \mathbb{R}^{n\times d_{in}} \to \mathbb{R}^{n\times d_{out}}$ and define a network function space $\mathcal{H}(L, \boldsymbol{p}, s, F, E, r)$ and Graph network $h \in \mathcal{H}(L, \boldsymbol{p}, s, F, E, r)$, where $h : \mathbb{R}^{n\times d_{in}} \to \mathbb{R}^{n\times d_{out}}$ is a mixed graph convolutional network and fully connected network with relu activation function. If we have n node features and node response $(X_i, Y_i), i = 1, 2..., n$ where $X_i \in \mathbb{R}^{d_{in}}, Y_i = g(X_{1:n})_i + \epsilon_i, Y_i \in d_{out}$ and h^* is the empirical minimizer of the equation $\sum_{i=1}^{n}(Y_i - h(X_{1:n})_i)^2$*

$$\mathbb{E}[(h^*(x) - g(x))^2] \leq C\phi_n L \log n^2$$

Where $\phi_n := \max_{i=0,\dots,q} n^{-\frac{2\beta_i^*}{\beta_i^*+t_i}}, \beta_i^* := \prod_{i+1}^{q}(\beta_l \wedge 1)$

Once we have the general form of the asymptotic analysis of the graph network, we can move to time evolving mechanism. Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and let $\mathbb{T} \subseteq \mathbb{R}$ be an discrete index set $\mathbb{T} \subseteq \{0, 1, 2, \dots\}$. A *filtration* is a family $\{\mathcal{F}_t\}_{t\in\mathbb{T}}$ of sub-σ-algebras of $\mathcal{F}$ satisfying $\mathcal{F}_s \subseteq \mathcal{F}_t$ whenever $s \leq t$.

Here the risk R is defined as

$$R(h^*, g, \mathbb{T}) = \mathbb{E}[\sum_{t\in\mathbb{T}} \mathbb{E}_{P_{X_t}}[h^*(X_t) - g(X_t)]^2 | \mathcal{F}_t] \tag{3.5}$$

Theorem 4. *Given Graph network $H = (h_1, h_2, h_3)$ and and $\hat{H}$ is the empirical minimizer of the equation $\sum_{t=1}^{T}\sum_{i=1}^{m_t}(X_{t,i} - H(\mathbf{X}_{t-1}, z_{1:m_t})_i)^2$, where $X_{t,i} = G(\mathbf{X}_{t-1}, z_{1:m_t})_i + \epsilon_{t,i}$ and prediction $\hat{X}_{t,i} = H(\mathbf{X}_t, z_{1:m_t})_i$ defined as $H(\mathbf{X}_{t-1}, z_{1:m_t}) = h_1(z_{1:m_t}) + h_2(\mathbf{X}_{t-1})$ for new points and $H(\mathbf{X}_{t-1}, z_{1:m_t}) = h_3(\mathbf{X}_{t-1})$ for existing points. Assume the new points set is not*

empty and with additional training on new points and updating of parametersψ_1, ψ_2, $\hat{H}$ *is now also the empirical minimizer of the equation* $\sum_{t=1}^{T}\sum_{i=m_t+1}^{m_t+n_t}(X_{t,i} - H(X_{t-1,1:m_t}, z_{1:m_t})_i)^2$. *Then there exists constant C, for any given non empty* $\mathbb{T} \subseteq \mathbb{Z}+$

$$R(G, \hat{H}, \mathbb{T})_{exist} \leq C\phi_{\sum_t m_t} L \log^2 \sum_{t=1}^{T} m_t \tag{3.6}$$

$$R(G, \hat{H}, \mathbb{T})_{new} \leq C\phi_{\sum_t n_t} L \log^2 \sum_{t=1}^{T} n_t \tag{3.7}$$

Where $R(G, \hat{H}, \mathbb{T})_{exist} = \mathbb{E}[\sum_{t\in\mathbb{T}} \mathbb{E}_{i\in V_{t-1}}[G(\mathbf{X}_{t-1}, z_{1:m_t})_i - H(\mathbf{X}_{t-1}, z_{1:m_t})_i]^2|\mathcal{F}_t]$ *and* $R(G, \hat{H}, \mathbb{T})_{new} = \mathbb{E}[\sum_{t\in\mathbb{T}} \mathbb{E}_{i\in I_t}[G(\mathbf{X}_{t-1}, z_{1:m_t})_i - H(\mathbf{X}_{t-1}, z_{1:m_t})_i]^2|\mathcal{F}_t]$ *Then for predictions, we have*

$$\mathbb{E}[\sum_{t\in\mathbb{T}} \mathbb{E}_{i\in I_t}[X_{t,i} - \hat{X}_{t,i}]^2|\mathcal{F}_t] \leq C\phi_{\sum_t m_t} L \log^2 \sum_{t=1}^{T} m_t + \sigma^2 \tag{3.8}$$

$$\mathbb{E}[\sum_{t\in\mathbb{T}} \mathbb{E}_{i\in V_{t-1}}[X_{t,i} - \hat{X}_{t,i}]^2|\mathcal{F}_t] \leq C\phi_{\sum_t n_t} L \log^2 \sum_{t=1}^{T} n_t + \sigma^2 \tag{3.9}$$

However, some graph networks do not meet the assumption, but according to the theorem 5, we can still get the convergence of the problem if the empirical loss converges.

Theorem 5. *Let h be a multilayer Graph neural network, and* $\mathcal{L} = \mathcal{L}(h, Y)$ *is the loss function with Lipchitz constant L, or in other words,* $h \in \mathcal{L}_{Lip,L}$. *Y is the target variable that h approximated. If we have iid sampled node feature and their target* $(X_i, Y_i), i = 1, 2..., m$.

$$\begin{aligned}\sup_{h\in\mathcal{L}_{Lip,L}} \left(R_{emp}(h, Y) - R_{exp}(h, Y)\right)^2 \leq \frac{2T8\|\mathcal{L}\|_\infty^2\pi}{m} + \\ \frac{2TL^2C}{m}\left(\frac{1}{N} + \frac{1+\log(N)}{N^{1/(D_{x_j}+1)}} + O\left(\exp(-N)N^{3T^{-1/2}}\right)\right),\end{aligned} \tag{3.10}$$

where $R_{emp}(\Theta, Y) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\Theta(X_i), Y_i)$ *is the empirical risk,* $R_{exp}(\Theta, Y) = \mathbb{E}_{X,Y}[\mathcal{L}(X, Y)]$ *is the expected risk on all possible nodes feature X and target Y, and the terms are defined as described in the context of the theorem.*

This theorem from paper [68] ensures the true loss will converge when the empirical loss converges. Then we have the theorem 6 bounding the real error by the empirical error.

Theorem 6. *For any given non empty* $\mathbb{T} \subseteq \mathbb{Z}+$ *and training data* $\hat{\mathbb{T}} \subseteq \mathbb{T}$*, given Graph network* $H = (h_1, h_2, h_3)$ *and and* $\hat{H}$ *is the empirical minimizer of the equation* $\sum_{t\in\hat{\mathbb{T}}} \sum_{i=1}^{m_t} (X_{t,i} - H(\mathbf{X}_{t-1}, z_{1:m_t})_i)^2$*, where* $X_{t,i} = G(\mathbf{X}_{t-1}, z_{1:m_t})_i + \epsilon_{t,i}$ *and prediction* $\hat{X}_{t,i} = H(\mathbf{X}_t, z_{1:m_t})_i$ *defined as* $H(\mathbf{X}_{t-1}, z_{1:m_t}) = h_1(z_{1:m_t}) + h_2(\mathbf{X}_{t-1})$ *for new points and* $H(\mathbf{X}_{t-1}, z_{1:m_t}) = h_3(\mathbf{X}_{t-1})$ *for existing points. Assume the new points set is not empty and with additional training on new points and updating of parameters*ψ_1, ψ_2*,* $\hat{H}$ *is now also the empirical minimizer of the equation* $\sum_{t\in\hat{\mathbb{T}}} \sum_{i=m_t+1}^{m_t+n_t} (X_{t,i} - H(\mathbf{X}_{t-1}, z_{1:m_t})_i)^2$*. Then there exists constant C,*

$$R(G, \hat{H}, \mathbb{T})_{new} \leq \frac{C}{|\sum_{t\in\hat{\mathbb{T}}} n_t|} + \hat{R}(G, \hat{H}, \hat{\mathbb{T}})_{new} + 2\sigma^2 \tag{3.11}$$

$$R(G, \hat{H}, \mathbb{T})_{exist} \leq \frac{C}{|\sum_{t\in\hat{\mathbb{T}}} m_t|} + \hat{R}(G, \hat{H}, \hat{\mathbb{T}})_{exist} + 2\sigma^2 \tag{3.12}$$

where empirical risk on existing points $\hat{R}(G, \hat{H}, \hat{\mathbb{T}})_{exist} = \frac{1}{|\hat{\mathbb{T}}|} \sum_{t\in\hat{\mathbb{T}}} \frac{1}{m_t} \sum_{i=1}^{m_t} [G(\mathbf{X}_{t-1}, z_{1:m_t})_i - H(\mathbf{X}_{t-1}, z_{1:m_t})_i]^2$ *and true risk on existing points* $R(G, \hat{H}, \mathbb{T})_{exist} = \mathbb{E}[\sum_{t\in\mathbb{T}} \mathbb{E}_{i\in V_{t-1}} [G(\mathbf{X}_{t-1}, z_{1:m_t})_i - H(\mathbf{X}_{t-1}, z_{1:m_t})_i]^2 | \mathcal{F}_t]$*, empirical risk on new points* $\hat{R}(G, \hat{H}, \hat{\mathbb{T}})_{new} = \frac{1}{|\hat{\mathbb{T}}|} \sum_{t\in\hat{\mathbb{T}}} \frac{1}{n_t} \sum_{i=m_t+1}^{m_t+n_t} [G(\mathbf{X}_{t-1}, z_{1:m_t})_i - H(\mathbf{X}_{t-1}, z_{1:m_t})_i]^2$ *and true risk on new points* $R(G, \hat{H}, \mathbb{T})_{new} = \mathbb{E}[\sum_{t\in\mathbb{T}} \mathbb{E}_{i\in I_t} [G(\mathbf{X}_{t-1}, z_{1:m_t})_i - H(\mathbf{X}_{t-1}, z_{1:m_t})_i]^2 | \mathcal{F}_t]$

3.5 Experiments

In this section, we will use the real-world data to justify and demonstrate the performance of our algorithm and compare our proposed algorithm with a variety of common existing algorithms.

3.5.1 Miami Housing

One dataset we use is the property price data in Miami [69], the data contains columns including the year of the property, the area of the property, the price, location(longitude and latitude), and distances to the closest stores, etc. Here the Price is used as $X_{t.i}$, which is the node prediction target while the year of the property is the time t, and the rest of the columns are features z_i.

The Figure 3.1 shows the location plot of the house price data. We can observe from the plot that most of the houses are condensed in blocks and communities. And by domain knowledge, we know the house prices in North America are highly similar in the same community. So it is very natural to use the graph network structure to model the house price according to the location of the property. For real data, we still assume that the hidden

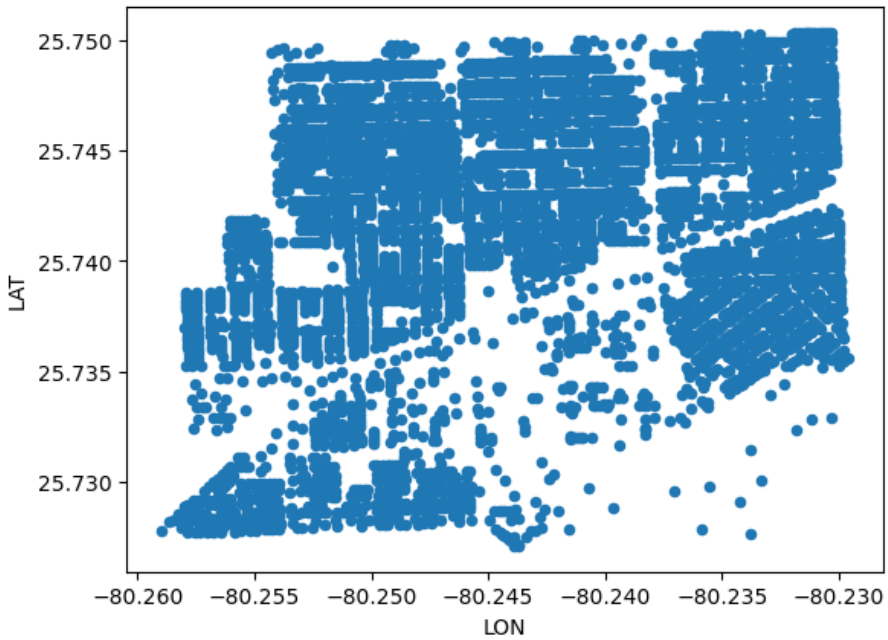


Figure 3.1: Location plot of house price data

function u exists from the neighborhood feature to the central node feature, but we will not include it explicitly in the algorithm. In our model, the graph structure of the data is determined by location. Essentially, the element of weighted adjacency matrix a_{ij} of node i

and j is calculated by $a_{ij} = exp^{-d_{ij}}$ and the distance is calculated by Haversine formula

$$d_{ij} = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\nu_j - \nu_i}{2}\right) + \cos(\nu_i)\cos(\nu_j)\sin^2\left(\frac{\lambda_j - \lambda_i}{2}\right)}\right) \tag{3.13}$$

where ν_i, λ_i are the latitude and longitude of the location.

And then we would like to demonstrate the model's capability in this dataset, and so we would like to use some other methodologies as comparisons. One of the comparison and benchmark we want to present is that this model can outperform the simple regression model. The simple regression from feature z to price $X_{t,i}$ does not provide an accurate prediction of the price. Meanwhile, the property price is variable along time t while the feature **z** is invariant to time, which does not provide enough information to update the price. We used Xgboost as the regressor here.

Arima We use the Auto-Arima function in Python as one of the benchmarks of our model. The function will automatically search for the best Seasonal Arima model on the data based on AIC criteria. The Seasonal Arima model, denoted as SARIMA$(p, d, q) \times (P, D, Q)_s$, extends the classical ARIMA framework by incorporating seasonal dynamics. Here, p, d, and q represent the orders of the non-seasonal autoregressive, differencing, and moving average components, respectively, while P, D, and Q denote the orders of the corresponding seasonal components, with s indicating the seasonal period. The general model formulation is expressed as:

$$\Phi_P(B^s)\,\phi_p(B)\,(1-B)^d(1-B^s)^D y_t = \Theta_Q(B^s)\,\theta_q(B)\,\varepsilon_t, \tag{3.14}$$

where B is the backshift operator such that $B^k y_t = y_{t-k}$, y_t is the observed time series at time t, $\phi_p(B)$ and $\theta_q(B)$ are the polynomials corresponding to the non-seasonal autoregressive and moving average terms, respectively, and $\Phi_P(B^s)$ and $\Theta_Q(B^s)$ are the polynomials for the seasonal autoregressive and moving average terms with seasonal lag s. The operator

$(1-B)^d(1-B^s)^D$ captures both non-seasonal and seasonal differencing to induce stationarity. The error term ε_t is assumed to be white noise.

Xgboost To demonstrate the importance of temporal and spatial information in our model, we used Xgboost as one of the benchmarks. The Xgboost only uses the feature information of each point to predict node response.

Temporal-spatial model with feature known Since the node features, including building type, distance to the store and etc., are usually known before the house is built in housing data. We would also like to demonstrate that the algorithm is capable of reaching a higher performance in predicting new nodes when features are already known. And we just need to adjust the feature extraction network h_1 to h_{ψ_1} to let it take node features of new points $z_{(m_t+1):(m_t+n_t)}$.

$$H_\psi(X_{t-1,1:m_t}, z_{1:m_t})_{t,(m_t+1):(m_t+n_t)} = h_{\psi_1}(z_{(m_t+1):(m_t+n_t)}) + h_2(X_{t-1,1:m_t})$$

Without node feature To demonstrate the necessity of node feature data, in the benchmark example only spatial, we exclude the node feature information from the following equation and compare the result with the original model

$$H_\psi(X_{t-1,1:m_t}, z_{1:m_t})_{t,(m_t+1):(m_t+n_t)} = h_2(X_{t-1,1:m_t})$$

However, if we only use the neighborhood price data, the training results are still not good enough as they lack the feature information to update and predict the price of existing points and new points.

The benchmark results (prediction loss in test dataset) are presented in the table 3.1

	Existing points	new points
Temporal-spatio model with feature known	**0.44**	**0.112**
Temporal-spatio model	**0.44**	**0.124**
Xgboost	0.46	0.46
Without node feature	**0.44**	0.126

Table 3.1: Benchmark with different algorithms

3.5.2 Air quality data in China

The second real dataset we use is the daily air quality data in China in 2018 [94]. The feature we include in this dataset is the daily weather data from 2018. Different from the housing data, air quality is seasonal, so we also include the seasonal temporal variable in the features, including day of the week, and month of the year. Similar to the housing price data, the original air quality data and weather data only has the Longitude and Latitude. We calculate the Euclidean distance by the Haversine formula for each pairs of points and get the adjacency matrix $A = (a_{ij}) = exp^{-d_{ij}}$.

The Figure 3.2 shows the location plot of air quality index data. We can observe from the plot that the air quality observation sites are not as condensed as the house price data and some of the cities are far away from others. The base map here is from openstreetmap [71]

Also, since the air pollution in China are usually related to the heating demand in cold time, the AQI data is seasonal as displayed in figure 3.3

As we want to demonstrate the model's performance in graphs with the emerging mechanism. The same to the house price data, we sampled 40% points at the start date 2018/01/01 as the initial data points and started from them. Each day, we sample additional 3-6 new data points. The figure 3.4 shows the sampling process of the first 4 days, including the initial sample, and presents them in different colors.

Since the graph networks are usually more than a single layer, the messages used to predict node responses are also not 1-Hop neighbor information; it has the capability to capture more distant node information. The figure 3.5 demonstrates the 1,2,3-hop neighbors

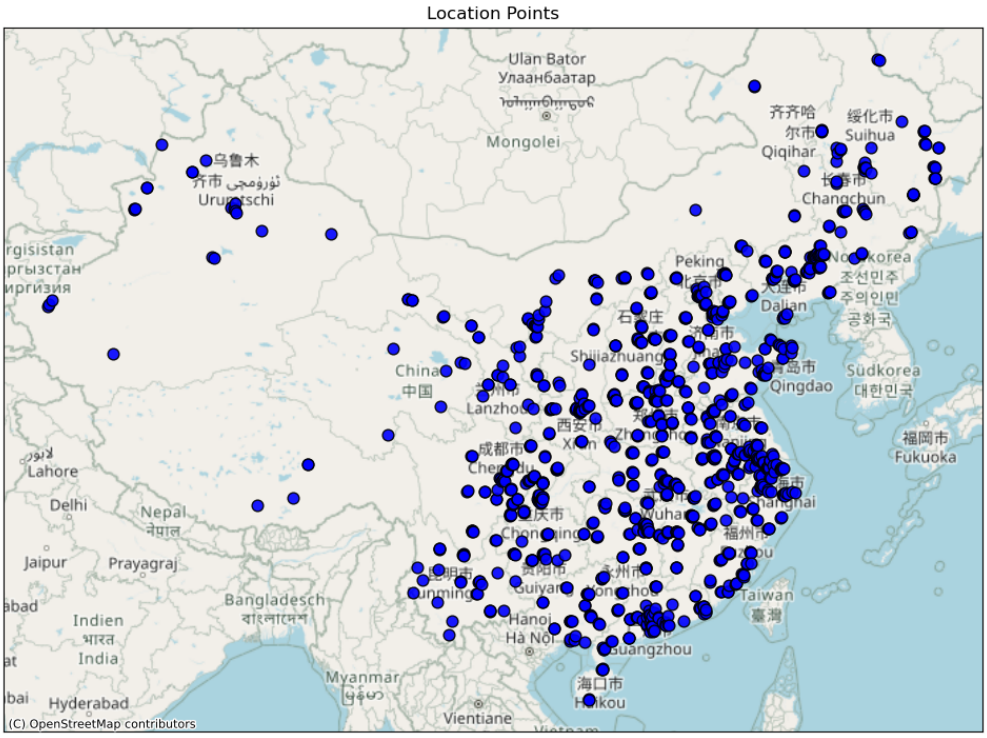


Figure 3.2: Location of observation sites in air quality index data

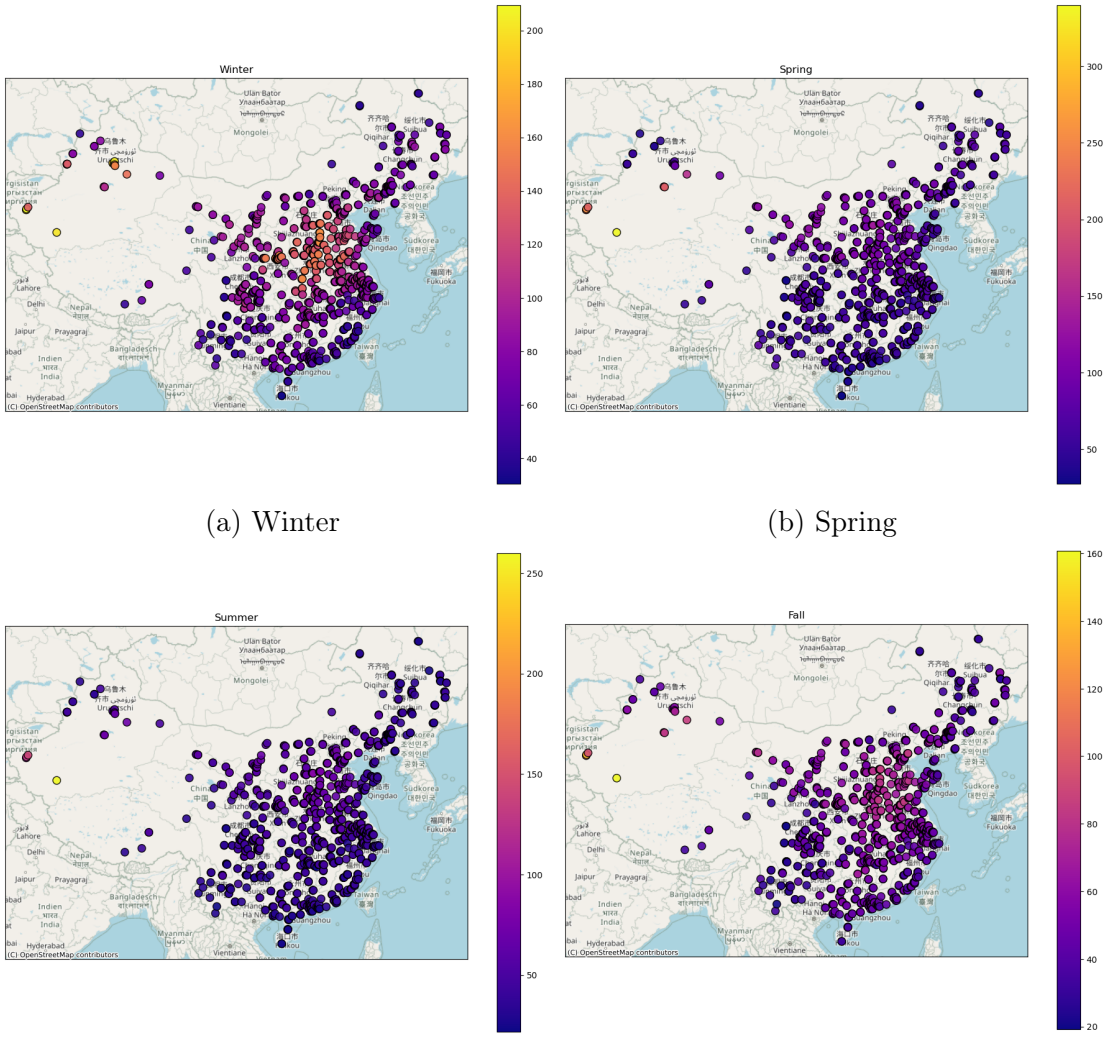
of some selected center points.

Here in the benchmark for AQI data, there is an additional **Temporal-spatial model on outliers** regime. Since the cities in China are not evenly distributed and some cities are far away from others, the prediction of AQI in such cities is usually not as accurate as in other cities, as they do not have enough neighboring nodes to calibrate the prediction. In the benchmark table 3.2, we displayed the results in such outlier cities as well.

The figure 3.6 also demonstrates how the location affects the prediction error.

	Existing points	**new points**
Temporal-spatio model	**0.33**	**0.27**
Temporal-spatio model on outliers	0.33	0.34
Seasonal Arima	0.35	NA
Xgboost	0.42	0.42
Without node feature	0.5	0.35

Table 3.2: Benchmark with different algorithms



(a) Winter

(b) Spring

(c) Summer

(d) Fall

Figure 3.3: AQI for different seasons.

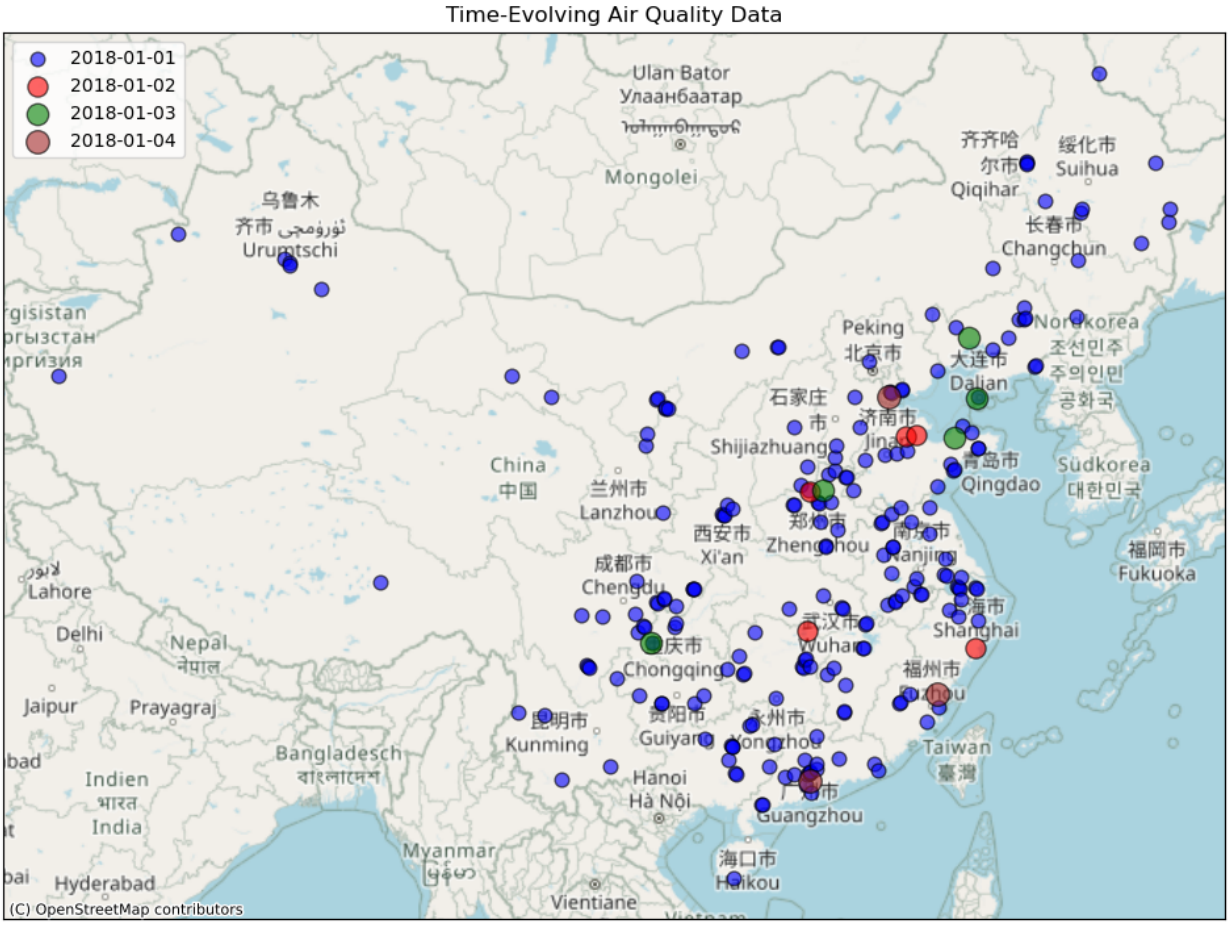


Figure 3.4: Emerging air quality data

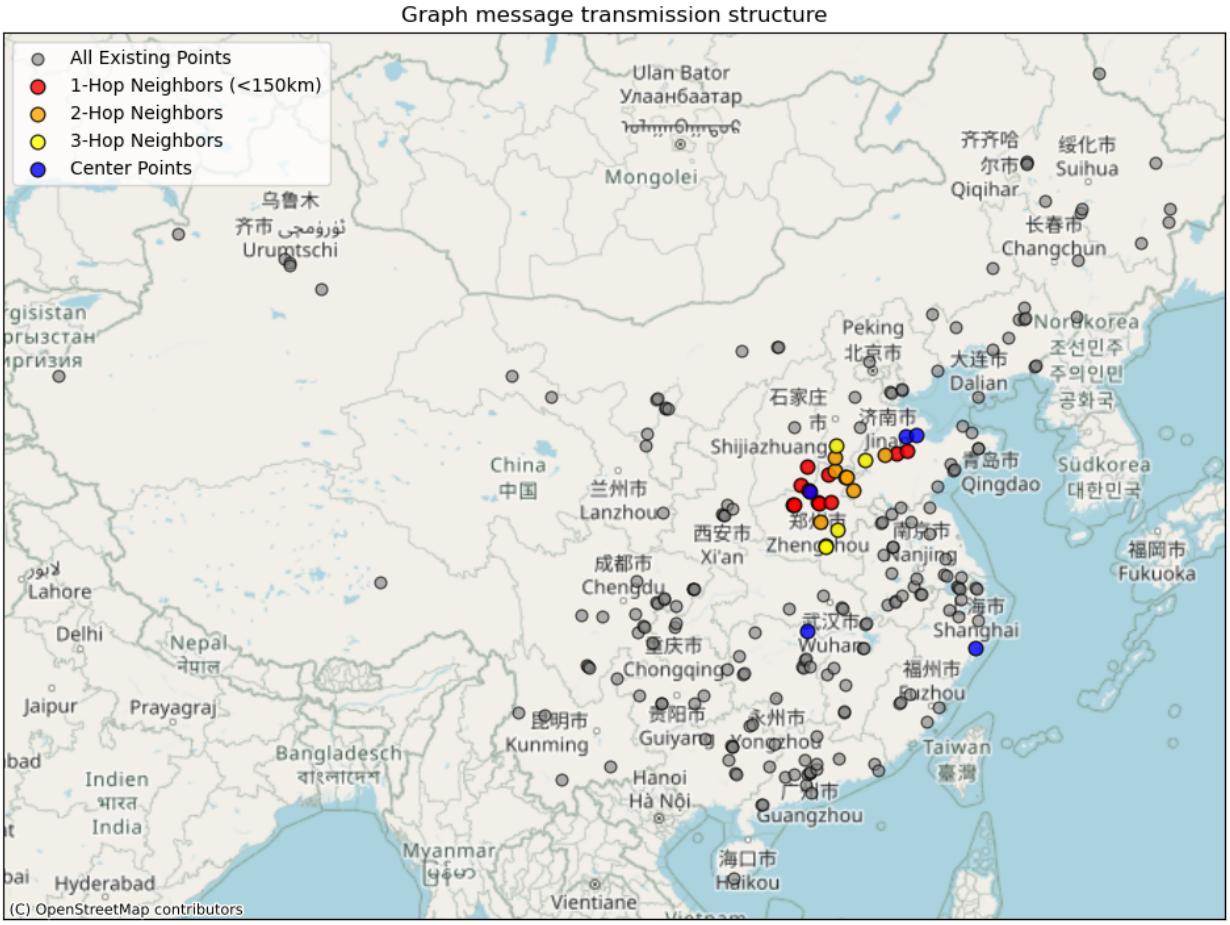


Figure 3.5: Message passing in air quality data

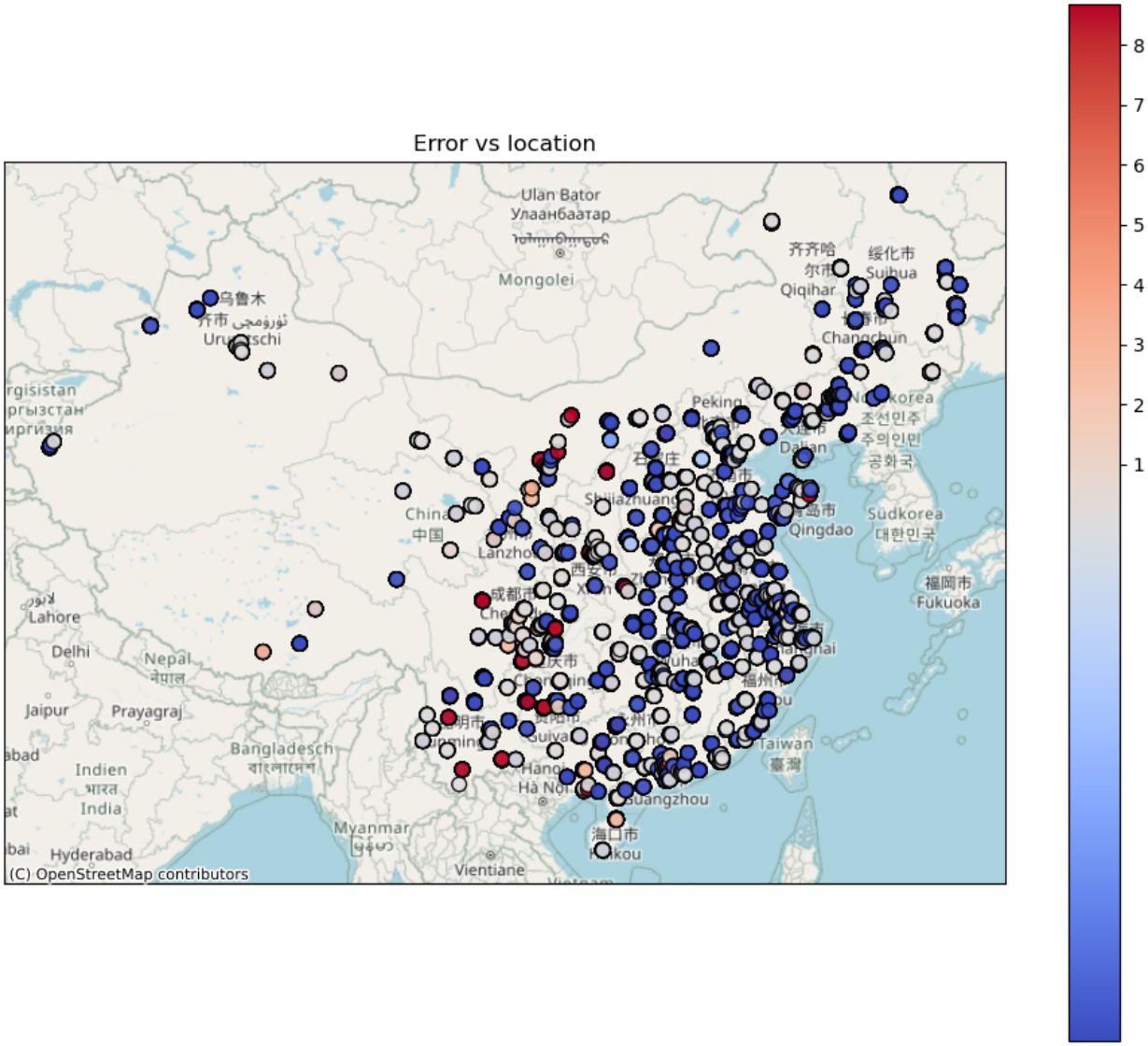


Figure 3.6: Relevant Error vs Geo-location

3.5.3 Data details

According to the algorithm, we can leverage the information from both feature and price data of neighborhood points and apply the graph neural network with hidden features. We start the training process by selecting a random set of properties as the initial data and keeping selecting random locations in the data as the newly built property for every year, representing the update of existing points and new points. Define the the node set as V and we have $V_1 \subset V_2 \subset \dots V_T = V$. We randomly sampled 40% of the dataset at $t = 1$ as initial existing node points set V_1, and at each time point t, we randomly sample 3-6 points from the dataset at time t, namely I_{t+1} defined in section 3.2.

For the Miami housing data, we select the housing price and features from 2010 to 2020 and use the first 8 years as a training set and the rest two years as a testing set. For Air quality data, we used the weather data, including precipitation, wind speed and etc, as node features and air quality data (in AQI index) as node responses. The data is from 360 days in 2018. We use the first 330 days as training data and the remaining 30 days as a testing dataset.

3.6 Proofs

3.6.1 Annotation

For any two number x and y, $x \gg y$ means x is much greater than y, $x \lesssim y$ means there exists a constant C unrelated to x and y, such that $x \leq Cy$ and $x \simeq y$ means there exists constant C_1, C_2 unrelated to x and y, such that $C_1 y \leq x \leq C_2 y$.

o(t) denotes $lim_{t\to 0}\frac{o(t)}{t} = 0$ when o(1) denotes $lim_{t\to 0} o(1) = 0$. For any real number x, $\lfloor x \rfloor$ denotes the largest integer strictly smaller than x.

For vector $v = (v_i)$ with dimension n, $|v|_1$ or $||v||_1$ is $\sum_{i=1}^{n} |v_i|$, $|v|_0$ or $||v||_0$ denotes the number of non-zero elements of v,$|v|_2$ or $||v||$ is $\sqrt{\sum_{i=1}^{n} v_i^2}$ and $|v|_\infty$ or $||v||_\infty$ is $\max\limits_{i=1,..,n} |v_i|$. For a $m \times n$ matrix $W = (W_{ij})$, $||W||_\infty = \max\limits_{i=1,..,n;j=1,...,m} |W_{ij}|$ and $||W||_0$ denotes the number of non-zero entries of W.

3.6.2 Assumption

Given model f_0, we assume it is measurable and composition of several functions where

$$g = k_q \circ k_{q-1} \circ ... \circ k_1 \circ k_0 \tag{3.15}$$

Here $k_i = (k_{ij})_j : [a_i, b_i]^{d_i} \to [a_{i+1}, b_{i+1}]^{d_{i+1}}$ while $k_{ij}^{-1}([a_{i+1}, b_{i+1}]) \subset [a_i, b_i]^{t_i} \subset [a_i, b_i]^{d_i}$ and $[a_i, b_i]^{t_i}$ to be domain of k_{ij}. Obviously, as $k_i = (k_{i0}, k_{i1}, ..., k_{id_{i+1}})$,$t_i \leq d_i$. Here $d_0 = p$ and $d_{q+1} = h$.

3.6.3 Lemma

First, define a type of function called β-Hölder smoothness whose partial derivatives exist up to $\lfloor \beta \rfloor$ order and are bounded and the partial derivatives of order $\lfloor \beta \rfloor$ are $\beta - \lfloor \beta \rfloor$ Hölder.

Then define the ball of β-Hölder smoothness functions with radius K as:

$$\mathcal{C}_\eta^\beta(D,K) = \{f : D \subset R^\eta \to R; \sum_{\boldsymbol{\alpha}:|\boldsymbol{\alpha}|<\beta} ||\partial^{\boldsymbol{\alpha}} f||_\infty + \sum_{\boldsymbol{\alpha}:|\boldsymbol{\alpha}|=\lfloor\beta\rfloor} \sup_{x,y\in D;x\neq y} \frac{|\partial^{\boldsymbol{\alpha}} f(x) - \partial^{\boldsymbol{\alpha}} f(y)|}{|x-y|_\infty^{\beta-\lfloor\beta\rfloor}} \leq K\}$$

where $\partial^{\boldsymbol{\alpha}} = \partial^{\alpha_1}...\partial^{\alpha_r}$ with $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_r) \in \mathbb{N}^r$ where ∂^{α_i} denotes the α_i partial derivative on the i^{th} dimension and $|\boldsymbol{\alpha}| := |\boldsymbol{\alpha}|_1$. We assume g_{ij} is β-Hölder smoothness then we define a function space to be:

$$\begin{aligned}\mathcal{G}(q, \boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\beta}, K, E) := \{&f = A_E k_q \circ A_E k_{q-1} \circ ... \circ A_E k_1 \circ A_E k_0 : \\ &k_i = (k_{ij})_j : [a_i, b_i]^{d_i} \to [a_{i+1}, b_{i+1}]^{d_{i+1}}, k_{ij} \in \mathcal{C}_{t_i}^{\beta_i}([a_i, b_i]^{t_i}, K) \\ &\text{where } |a_i| \leq K, |b_i| \leq K, A_E = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}\}\end{aligned} \tag{3.16}$$

where $\boldsymbol{d} := (d_0, ..., d_{q+1}), \boldsymbol{t} := (t_0, ..., t_q), \boldsymbol{\beta} := (\beta_0, ..., \beta_q)$, $\tilde{A}, \tilde{D}^{-\frac{1}{2}}$ is the normalized adjacency matrix and degree matrix according to weighted edge E respectively.

Then we start to define the function space of network functions. First, we define a network function space with bounded parameters as:

$$\begin{aligned}\mathcal{H}(L, \boldsymbol{p}, E, r) := \{&f : \mathbb{R}^{p_0} \to \mathbb{R}^{p_{L+1}}, x \to f(x) = l_r \circ l_{r-1} ... \circ l_1 : \\ &l_i(x) = A_E W_{L_{l_i}} \sigma_{v_{L_{l_i}}} W_{L_{l_i}-1} \sigma_{v_{L_{l_i}-1}} ... W_{L_{l_{i-1}+1}} x : \\ &i = 1, ...r, l_0 = 0, l_r = L, \max_{j=0,...,L} ||W_j||_\infty \vee |v_j|_\infty \leq 1, L_{l_r} = L, L_{l_1} = 0\}\end{aligned} \tag{3.17}$$

Where $\boldsymbol{p} = (p_0, ..., p_{L+1}) \in \mathbb{N}^{L+2}$, L is number of layers of the network, W_j is the $p_{j+1} \times p_j$ matrix, $v_j \in \mathbb{R}^{p_j}$ and $\sigma_{v_j}(x) = \max(0, x - v_j)$ is the ReLU function. Then $f : \mathbb{R}^{p_0} \to \mathbb{R}^{p_{L+1}}$ defined above is a neuro network with ReLU activation. Here $p_0 = d_{in}$ and $p_{L+1} = d_{out}$.

Let $|||f|_\infty||_\infty$ be sup-norm of the function $x \to |f(x)|_\infty$. Add the sparse condition to the

parameters and define the network function space as:

$$\mathcal{H}(L, \boldsymbol{p}, s, E, r) := \mathcal{H}(L, \boldsymbol{p}, s, F, E, r) := \\ \{f \in \mathcal{H}(L, \boldsymbol{p}, E) : \sum_{j=0}^{L} ||W_j||_0 + |v_j|_0 \leq s, |||f|_\infty||_\infty \leq F\} \tag{3.18}$$

Then, start to approximate the convergence rate. First, let

$$\beta_i^* := \prod_{i+1}^{q} (\beta_l \wedge 1) \tag{3.19}$$

And define convergence rate as:

$$\phi_n := \max_{i=0,\ldots,q} n^{-\frac{2\beta_i^*}{\beta_i^* + t_i}} \tag{3.20}$$

where n is the size of the training set where the network is trained.

Lemma 1. *Let $g \in \mathcal{G}(q, \boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\beta}, K, E)$ and define a network function space $\mathcal{H}(L, \boldsymbol{p}, s, F, E, r)$.Let $Y_i = g(X_{1:n})_i + \epsilon_i$ as defined above and$\hat{h}_n \in argmin_{h \in \mathcal{H}(L,p,s,F,r)} \sum_{i=1}^{n} (Y_i - h(X_{1:n})_i)^2$be an empirical risk minimizer and satisfied conditions below:*

(i)$F \geq \max(K, 1)$

(ii)$\sum_{i=0}^{q} log_2(4t_i \vee 4\beta_i) log_2 n \leq L \lesssim n\phi_n$

(iii)$n\phi_n \lesssim \min_{i=1,\ldots,L} p_i$

(iv)$s \simeq n\phi_n logn$

There exists a constant C' only depending on q,$\boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\beta}$*,F such that*

$$R(\hat{h}_n - g) = E_g[(\hat{h}_n(X) - g(X))^2] \leq C' \phi_n L log^2 n$$

3.6.4 Proof of Lemma 1 or Theorem 3

Lemma 2. *For any function $g \in C_d^\beta([0,1]^d, K)$ and any integers $m \geq 1$ and $N \geq (\beta+1)^d \vee (K+1)e^d$, there exists a network*

$$\tilde{h} \in \mathcal{H}\left(L, (d, 6(d+\lceil\beta\rceil)N, \ldots, 6(d+\lceil\beta\rceil)N, 1), s, \infty\right),$$

with depth

$$L = 8 + (m+5)(1 + \lceil \log_2(d \vee \beta)\rceil),$$

and number of parameters

$$s \leq 141(d+\beta+1)^{3+d} N(m+6),$$

such that

$$\|\tilde{h} - g\|_{L^\infty([0,1]^d)} \leq (2K+1)(1+d^2+\beta^2)6^d N 2^{-m} + K 3^\beta N^{-\frac{\beta}{d}}.$$

Proof of lemma 2

Based on the proof in [80] and add the graph structure A_E

Given a function $g \in \mathcal{G}(q, \boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\beta}, K, E)$, then $f = A_E k_q \circ A_E k_{q-1} \circ ... \circ A_E k_1 \circ A_E k_0$. For each of the component $A_E k_i$, where $k_i = (k_{ij})_j, k_{ij} \in \mathcal{C}_{t_i}^{\beta_i}([a_i, b_i]^{t_i}, K_{ij})$. Then use the normalization trick

$$c_0 := \frac{k_0}{2K_0} + \frac{1}{2}, \quad c_i := \frac{k_i(2K_{i-1}x - K_{i-1})}{2K_i} + \frac{1}{2}, \quad c_q = k_q(2K_{q-1}x - K_{q-1}).$$

where $c_i = (c_{ij})_j$ and $2K_{i-1}x - K_{i-1}$ means that we transform the entries by $2K_{i-1}x_j - K_{i-1}$ for all j. Clearly,

$$g = A_E k_q \circ \cdots \circ A_E k_0 = A_E c_q \circ \cdots \circ A_E c_0. \tag{3.21}$$

Using the definition of the Hölder balls $C_d^\beta(D,K)$, it follows that $c_0 j$ takes values in $[0,1]$,

$$c_{0j} \in C_{t_0}^\beta([0,1]^{t_0},1), \quad c_{ij} \in C_{t_i}^\beta([0,1]^{t_i},(2K_{i-1})^{\beta_i}) \quad \text{for } i=1,\ldots,q-1,$$

and

$$c_{qj} \in C_{t_q}^\beta([0,1]^{t_q},K_q(2K_{q-1})^{\beta_q}).$$

Without loss of generality, we can always assume that the radii of the Hölder balls are at least one, that is, $K_i \geq 1$.'

Then by the above lemma, there exists a network

$$\tilde{h} \in \mathcal{H}\left(L,(d,6(d+\lceil\beta\rceil)N,\ldots,6(d+\lceil\beta\rceil)N,1),s,\infty\right),$$

with depth

$$L = 8+(m+5)(1+\lceil\log_2(d\vee\beta)\rceil),$$

and number of parameters

$$s \leq 141(d+\beta+1)^{3+d}N(m+6),$$

such that

$$\|\tilde{h}-c_{ij}\|_{L^\infty([0,1]^d)} \leq (2K+1)(1+d^2+\beta^2)6^dN2^{-m}+K3^\beta N^{-\frac{\beta}{d}}.$$

Then

$$\|A_E\tilde{h}-A_Ec_{ij}\|_{L^\infty([0,1]^d)} \leq (2K+1)(1+d^2+\beta^2)6^dN2^{-m}+K3^\beta N^{-\frac{\beta}{d}}.$$

because A_E is the normalized weights.

Then given any component of function $g \in \mathcal{G}(q,\boldsymbol{d},\boldsymbol{t},\boldsymbol{\beta},K,E)$ k_i,

there exists $\tilde{h} \in \mathcal{H}\left(L,(d,6(d+\lceil\beta\rceil)N,\ldots,6(d+\lceil\beta\rceil)N,1),s,\infty,E\right)$,

$$\|A_E\tilde{h}-A_Ek_{ij}\|_{L^\infty([0,1]^d} \leq (2K+1)(1+d^2+\beta^2)6^dN2^{-m}+K3^\beta N^{-\frac{\beta}{d}}.$$

where $k_i = (k_{ij})_j$.

Following the proof in the supplement of paper, the lemma is proved.

3.6.5 Main proof of Theorem 4

$$\text{If } h_3^* = \arg\min_{h_3} \sum_{t=1}^{T} \sum_{i=1}^{m_t} (X_{t,i} - h_3(\mathbf{X}_{t-1})_i)^2$$

Then

$$R(g_3, h_3^*, \mathbb{T})_{exist} = \mathbb{E}[\sum_{t\in\mathbb{T}} \mathbb{E}_{i\in V_{t-1}}[(g_3(\mathbf{X}_{t-1})_i - h_3(\mathbf{X}_{t-1})_i)^2 | \mathcal{F}_t]] \leq C\phi_{\sum_{t=1}^{T} m_t} L \log^2(\sum_{t=1}^{T} m_t)$$

If

$$H^* = \arg\min_{H} \sum_{t=1}^{T} \sum_{i=1}^{n_t} (X_{t,i} - H(\mathbf{X}_{t-1}, z_{1:m_t})_i)^2$$

Define

$$h_1^* = \arg\min_{h_1} \sum_{t=1}^{T} \sum_{i=1}^{n_t} (g_1(z_{1:m_t})_i - h_1(z_{1:m_t})_i + \epsilon_{t,i})^2$$

and

$$h_2^* = \arg\min_{h_2} \sum_{t=1}^{T} \sum_{i=1}^{n_t} (g_2(\hat{X}_{t,1:m_t}) - h_2(\hat{X}_{t,1:m_t}) + \epsilon_{t,i})^2$$

where

$$\hat{X}_{t,1:m_t} = h_3^*(\mathbf{X}_{t-1}) \text{ let } H^{**}_{i\in V_{t-1}} = \hat{X}_{t,1:m_t}$$

Let

$$H^{**} \text{ be the function } \hat{X}_{t,m_t+1:m_t+n_t} = H^{**}(\mathbf{X}_{t-1}, z_{1:m_t}) = h_1^*(z_{1:m_t}) + h_2^*(h_3^*(\mathbf{X}_{t-1}))$$

for new nodes,then

$$R(G, H^*, \mathbb{T}) \leq R(G, H^{**}, \mathbb{T}) \text{and for } h_1^*, R(h_1^*, g_1, \mathbb{T}) \leq C'\phi_{\sum_{t=1}^{T} n_t} L log^2 \sum_{t=1}^{T} n_t, \text{ for } h_2^*,$$

$$R(h_2^*, g_2, \mathbb{T}) \leq C' \phi_{\sum_{t=1}^T n_t} Llog^2 \sum_{t=1}^T n_t \text{ then } R(G, H^*, \mathbb{T})_{new} \leq C \phi_{\sum_{t=1}^T n_t} Llog^2 \sum_{t=1}^T n_t$$

$$\mathbb{E}[\mathbb{E}_{i \in I_t}(\hat{X}_{t,i} - X_{t,i})^2 | \mathcal{F}_t] \leq R(G, \hat{H}, \mathbb{T})_{new} + \mathbb{E}_{t \in \mathbb{T}, i \in I_t}(\epsilon_{t,i}^2) \leq \sigma^2 + C' \phi_{\sum_{t=1}^T n_t} Llog^2 \sum_{t=1}^T n_t$$

$$\mathbb{E}[\mathbb{E}_{i \in V_{t-1}}(\hat{X}_{t,i} - X_{t,i})^2 | \mathcal{F}_t] \leq R(G, \hat{H}, \mathbb{T})_{exist} + \mathbb{E}_{t \in \mathbb{T}, i \in V_{t-1}}(\epsilon_{t,i}^2) \leq \sigma^2 + C' \phi_{\sum_{t=1}^T m_t} Llog^2 \sum_{t=1}^T m_t$$

3.6.6 Proof of extended theorems: Theorem 5 and Theorem 6

Let $G = (V, E)$ be a weighted graph and g be a MPNN [68]. For each $t \in \{1, \ldots, T\}$, we define the MPNN $\Theta_G^{(t)}$ as the mapping that maps input graph signals $f = f^{(0)} \in \mathbb{R}^{N \times F_0}$ to the features in the t-th layer by

$$\Theta_G^{(t)} : \mathbb{R}^{N \times F_0} \to \mathbb{R}^{N \times F_t}, \quad f \mapsto f^{(t)} = \big(f_i^{(t)}\big)_{i=1}^N,$$

where $f^{(t)} \in \mathbb{R}^{N \times F_t}$ are defined sequentially by

$$m_i^{(t)} := \frac{1}{d_i} \sum_{j=1}^N w_{i,j} \Phi^{(t)}\big(f_i^{(t-1)}, f_j^{(t-1)}\big),$$

$$f_i^{(t)} := \Psi^{(t)}\big(f_i^{(t-1)}, m_i^{(t)}\big),$$

for every $i \in V$. We call $\Theta_G := \Theta_G^{(T)}$ a message passing graph neural network (gMPNN).

Given a MPNN Θ as defined in Definition 2.1, the output $\Theta_G(f) \in \mathbb{R}^{N \times F_T}$ is a graph signal. In graph classification or regression, the network should output a single feature for the whole graph. Hence, the output of a gMPNN after *global pooling* is a single vector $\Theta_G^P(f) \in \mathbb{R}^{F_T}$, defined by

$$\Theta_G^P(f) = \frac{1}{N} \sum_{i=1}^N \Theta_G(f)_i.$$

Let $g = g_{\phi,\psi,W}$ and Let W=$\begin{pmatrix} a & 0 \\ 0 & W_z \end{pmatrix}$ where W_z is the weight matrix for feature z_i. Here define the $\phi(x) = x^T W$ and $\psi(x) = relu(x)$.

For new points $\mathbb{E}[\mathbb{E}_{i\in I_t}(\hat{X}_{t,i} - X_{t,i})^2|\mathcal{F}_t] = \mathbb{E}[\mathbb{E}_{i\in I_t}[(h_1(z_{1:m_t})_i + h_2(\hat{X}_{t,1:m_t})_i - X_{t,i})^2|\mathcal{F}_t]$

$$\leq \mathbb{E}[\mathbb{E}_{i\in I_t}[(g_1(z_{1:m_t})_i - h_1(z_{1:m_t})_i)^2|\mathcal{F}_t] + \mathbb{E}[\mathbb{E}_{i\in I_t}[(g_2(\hat{X}_{t,1:m_t})_i - h_2(\hat{X}_{t,1:m_t})_i)^2|\mathcal{F}_t] + E(\epsilon_{t,i}^2)$$

$$\leq \frac{1}{m}\sum_{i=1}^{m}(h_1(z_{1:m_t})_i + h_2(\hat{X}_{t,1:m_t})_i - X_{t,i})^2 + o(\frac{1}{m}) + 2\sigma^2$$

For existing points $\mathbb{E}[\mathbb{E}_{i\in V_{t-1}}(\hat{X}_{t,i} - X_{t,i})^2|\mathcal{F}_t] = \mathbb{E}[\mathbb{E}_{i\in V_{t-1}}[h_3(\hat{X}_{t,1:m_t})_i - X_{t,i})^2|\mathcal{F}_t]$

$$\leq \mathbb{E}[\mathbb{E}_{i\in V_{t-1}}[(g_3(\mathbf{X}_{t-1})_i - h_3(\mathbf{X}_{t-1})_i)^2|\mathcal{F}_t] + E(\epsilon_{t,i}^2)$$

$$\leq \frac{1}{m}\sum_{i=1}^{m}(h_3(\mathbf{X}_{t-1})_i - X_{t,i})^2 + o(\frac{1}{m}) + 2\sigma^2$$

Chapter 4

A two-step neural-network based nonparametric estimator with nuclear penalization and network output truncation robust to different data contamination mechanisms

4.1 Introduction

Data corruption mechanisms of interest are due (i) to happenstance - for example, small distinct subpopulations of data are incorporated in the core data set [59, 75] or measurements are incorrectly recorded due to device malfunction [56, 96], erroneous record keeping [36, 81], or due to self-reported inaccuracies [30, 76]- and (ii) to malicious activity by adversaries (see, e.g., [1, 5, 24, 31, 83, 88, 101] and references therein). The learning task of interest is that of *estimation* of the *parameter* of a predictive model (regression or classification).

While the field of *robust statistics* has explored contamination for fixed-dimension models

[40, 41, 45–48, 67, 78], robustness in high-dimensional settings remains underexplored despite its critical importance in large-data applications.

To elucidate the key issues and ideas, consider a data set $\mathcal{D} = \{(y_i, x_i)\}_{i=1}^n$, wherein $y_i \in \mathbb{R}$ is a response/outcome and $\boldsymbol{x}_i \in \mathbb{R}^p$ a predictor vector corresponding to the i^{th} observation. One is interested in estimating the parameter $\beta \in \mathbb{R}^p$ of the following *linear* regression model: $y_i = x_i^T \beta_{\text{true}} + \varepsilon_i$ (more complex predictive models are discussed in the sequel). The statistical and machine learning literature on robust regression largely focuses on the following data contamination mechanisms:

• **Error/Informative contamination model.** Observations $\{(y_i, x_i)\}_{i=1}^n$ are identically distributed with F_{true} (described next) as the common distribution. We posit $(y, x) \sim F_{\text{true}}$, if $y = x^T \beta_{\text{true}} + \varepsilon$, where x and ε both have mean zero, and are *independent*. The common error distribution under F_{true}, denoted by $F_{\varepsilon,\text{true}}$, is assumed to be a heavy-tailed distribution. Such heavy tailed errors commonly arise in actuarial and financial applications [74]. The classical contamination model of Huber [41] is also a special case when $F_{\varepsilon,\text{true}}$ is a discrete mixture of a normal distribution and another heavy tailed distribution. Observations with heavy-tailed errors are considered *contaminated or outliers*. While the signal $x^T \beta_{\text{true}}$ remains intact, the noise is amplified. These observations still contain useful information about β_{true} and should be adjusted for proper estimation.

• **Strong/Non-informative contamination model.** Observations $\{(y_i, x_i)\}_{i=1}^n$ are identically distributed, where the common distribution can be described as follows: with probability $1 - \delta$, $(y_i, x_i) \sim F_{\text{true}}$, with ϵ_i following a sub-Gaussian distribution, while with probability $\delta > 0$, (y_i, x_i) follow an *arbitrary* contamination distribution denoted by F_{cont}. In this setting, contaminated observations may have errors dependent on the predictors and, in extreme cases, provide no information about β_{true}, e.g., if $y = 0$ (or $\epsilon = -x^T \beta_{\text{true}}$). Predictor values may also be corrupted.

The first mechanism can arise when data corruption occurs due to happenstance, while the latter one when data corruption occurs due to malicious activity, as discussed above.

4.2 Problem formulation

Annotation Given r dimension vector $y = \{y_i\}_{i=1}^r \in \mathbb{R}^r$, $\sigma_v y = \sigma_v(y) = \begin{pmatrix} \max(y_1 - v_1, 0) \\ \max(y_2 - v_2, 0) \\ \cdots \\ \max(y_r - v_r, 0) \end{pmatrix}$, and $\sigma = \sigma_0$ represents the ReLu activation function. Define nuclear norm of a matrix A as $\|A\|_* = \sum_i \sigma_i(A)$ where σ_i is ith singular value of matrix A. Given any real number a, $\lfloor a \rfloor$ represents the largest integer that is less than or equal to a. $\mathbb{N}$ represents integer set.

Consider a regression problem with total time points T and input dimension d_{in} and output dimension d_{out} and with error distribution $Q_i, i = 1, 2, \ldots T$ with contamination rate ϵ. Here the distribution Q_i is heavy-tailed or even problematic, which means $\mathbb{E}_{x \sim Q_i}[e^{tx}] = \infty$ with probability ϵ, given a sufficiently small error parameter quantifying the magnitude of contamination δ

Let's define a function space we will use later, this function space can be found on [80]. Define the ball of β-Hölder functions with radius K as

$$\mathcal{C}_d^\beta(D, K) = \left\{ f : D \subset \mathbb{R}^d \to \mathbb{R} \;\middle|\; \sum_{\alpha:|\alpha|<\beta} \|\partial^\alpha f\|_\infty + \sum_{\alpha:|\alpha|=\lfloor\beta\rfloor} \sup_{\substack{\mathbf{x},\mathbf{y}\in D \\ \mathbf{x}\neq\mathbf{y}}} \frac{|\partial^\alpha f(\mathbf{x}) - \partial^\alpha f(\mathbf{y})|}{\|\mathbf{x} - \mathbf{y}\|_\infty^{\beta - \lfloor\beta\rfloor}} \leq K \right\},$$

where $\partial^\alpha = \partial^{\alpha_1} \cdots \partial^{\alpha_d}$ for multi-index $\alpha = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}^d$ and $|\alpha| := \sum_{i=1}^d \alpha_i$. This property is similar to Lipchitz.

We then assume the true model function f is the composition of such kind of β-Hölder functions:

$$f = g_q \circ g_{q-1} \circ \cdots \circ g_1 \circ g_0,$$

where

$$g_\ell : [\mathbf{a}_\ell, \mathbf{b}_\ell]^{d_\ell} \to [\mathbf{a}_{\ell+1}, \mathbf{b}_{\ell+1}]^{d_{\ell+1}}, \quad g_\ell = (g_{\ell j})_{j=1}^{d_{\ell+1}}, \quad \ell = 0, \ldots, q,$$

with unknown parameters d_ℓ and q. Here $d_0 = d_{in}, d_q = d_{out}$ represents the input and output

dimension of model function.

Assume each $g_{\ell j}$ is β_ℓ-Hölder with radius K_ℓ. Let t_ℓ be the maximal number of variables on which each $g_{\ell j}$ depends, and suppose $t_\ell \leq d_\ell$. Since $g_{\ell j}$ is also t_ℓ-variate, so, we restrict the function to its true domain and the true underlying function space becomes

$$\mathcal{G}(q, \mathbf{d}, \mathbf{t}, \boldsymbol{\beta}, \mathbf{K}) := \Big\{ f = g_q \circ \cdots \circ g_0 \;\Big|\; g_\ell = (g_{\ell j})_{j=1}^{d_{\ell+1}}, \\ g_{\ell j} \in \mathcal{C}_{t_\ell}^{\beta_\ell}\left([\mathbf{a}_\ell, \mathbf{b}_\ell]^{t_\ell}, K_\ell\right),\ \|\mathbf{a}_\ell\|_\infty, \|\mathbf{b}_\ell\|_\infty \leq K_\ell \Big\}. \quad (4.1)$$

We define the parameter vectors as follows:

$$\mathbf{d} := (d_0, \ldots, d_{q+1}), \quad \mathbf{t} := (t_0, \ldots, t_q), \quad \boldsymbol{\beta} := (\beta_0, \ldots, \beta_q), \quad \mathbf{K} := (K_0, \ldots, K_q),$$

and define the effective smoothness index by

$$\beta_\ell^* := \beta_\ell \prod_{k=\ell+1}^{q} (\beta_k \wedge 1).$$

Now let's define the data contamination model as

$$Y_i = f(X_i) + Q_i, i = 1, 2, ..., T, \mathbb{E}[Q_i] \leq \delta, X_i \in \mathbb{R}^{d_{in}}, Y_i \in \mathbb{R}^{d_{out}} \quad (4.2)$$

where ϵ represent the rate of contamination and $f : \mathbb{R}^d \to \mathbb{R}^q$ and The true regression function $f \in \mathcal{G}(q, \mathbf{d}, \mathbf{t}, \boldsymbol{\beta}, \mathbf{K})$.

Define the dataset $\mathcal{D}_\mathcal{T} = ((X_i, Y_i), i = 1, 2, \ldots, T)$ where X_i represents the input data and Y_i represents the output data.

4.3 Proposed Method

We will propose a two-step machine learning methodology. The first step is to approximate the main part of the model function f by a neural network with nuclear penalization. Define neural network $\hat{f} = \mathbf{W}_L\sigma\left(\mathbf{W}_{L-1}\cdots\sigma\left(\mathbf{W}_1\sigma\left(\mathbf{W}_0 x\right)\right)+\cdots\right), x \in \mathbb{R}^{d_{in}}$, where $F_l = W_l\sigma(W_{l+1}\sigma(...W_L x))$. Note that this network does not contain any bias vectors. Given the batch size B, we group the data in batch $(X_{(j-1)B:iB}, Y_{(j-1)B:iB}), j = 1, 2, \ldots \lfloor\frac{T}{B}\rfloor$ where $X_{(j-1)B:iB} = (X_{(j-1)B}, X_{(j-1)B+1}, \ldots, X_{jB-1})$ and $Y_{(j-1)B:iB} = (Y_{(j-1)B}, Y_{(j-1)B+1}, \ldots, Y_{jB-1})$. Instead of penalizing the nuclear norm of the weight matrix, we actually penalize the nuclear norm of each layer output F_l. Note that here the layer output is based on each batch. We set the loss function $\mathcal{L} = \mathcal{L}_\theta = \|Y_{(j-1)B:jB} - \hat{f}(X_{(j-1)B:jB})\| + \sum_{l=1}^{L}||F_l||_*$.

Algorithm 4 Nuclear penalization step

1: **System Initialization**
2: Input data X_i and output data Y_i $i = 1, 2, \ldots$,number of epochs N and number of points T,batch size B, neural network function $\hat{f}$ with trainable parameter θ.
3: **for** *epoch* $= 1, 2, \ldots, N$ **do**
4: **for** $j = 1, 2, \ldots, \lfloor\frac{T}{B}\rfloor$ **do**
5: Compute the loss:

$$\mathcal{L}_\theta = \|Y_{(j-1)B:jB} - \hat{f}(X_{(j-1)B:jB})\| + \sum_{l=1}^{L}||F_l||_*$$

6: **while** θ not converged **do**
7: Update parameters: $\theta \leftarrow$ ADAM($\mathcal{L}_\theta$)
8: **end while**
9: **end for**
10: **end for**
11: **Output** Trained network function $\hat{f}$ with parameter $\hat{\theta}$

Then we will define a special feed-forward network space. For an integer $L \geq 1$ and $\mathbf{p} = (p_0, p_1, \ldots, p_L, p_{L+1}) \in \mathbb{N}^{L+2}$, let $\mathcal{F}(L, \mathbf{p})$ denote the class of DNNs with L hidden layers and p_ℓ nodes on hidden layer ℓ, for $\ell = 1, \ldots, L$, and any $f \in \mathcal{F}(L, \mathbf{p})$ has a composition

structure:

$$f(\mathbf{x}) = \mathbf{W}_L \sigma\left(\mathbf{W}_{L-1} \cdots \sigma\left(\mathbf{W}_1 \sigma\left(\mathbf{W}_0 \mathbf{x} + \mathbf{u}_0\right) + \mathbf{u}_1\right) + \cdots + \mathbf{u}_{L-1}\right) + \mathbf{u}_L, \quad \mathbf{x} \in \mathbb{R}^{d_{in}} \tag{4.3}$$

where $\mathbf{W}_\ell \in \mathbb{R}^{p_{\ell+1} \times p_\ell}$ are weight matrices and $\mathbf{u}_\ell \in \mathbb{R}^{p_{\ell+1}}$ are shift vectors for $\ell = 0, \ldots, L$.

The following function space is based on $\mathcal{F}(L, \mathbf{p})$ and with sparsity penalization to prevent overfitting.

$$\mathcal{F}(L, \mathbf{p}, s) = \{f \in \mathcal{F}(L, \mathbf{p}) : \sum_{\ell=0}^{L} \|\mathbf{W}_\ell\|_0 + \|\mathbf{u}_\ell\|_0 \leq s, \\ \max_{\ell=0,\ldots,L} \|\mathbf{W}_\ell\|_\infty \vee \|\mathbf{u}_\ell\|_\infty \leq 1, \ \|f\|_\infty \leq C\} \tag{4.4}$$

where $\|\cdot\|_\infty$ denotes the maximum-entry norm of a matrix or vector (i.e., sup norm), and $\|\cdot\|_0$ denotes the number of non-zero entries. The sparsity level $s > 0$ controls the number of nonzero weights and shift vectors. C controls the range of network f.

To simplify the notation, we write $\mathcal{F}(L, \mathbf{p}, s)$ as simply $\mathcal{F}$ in what follows.

To further prevent overfitting, we add another parameter to this function space γ to control the maximum of the L_2 norm of the weight matrix.

$$\mathcal{F}(L, \mathbf{p}, s, \gamma) = \{f \in \mathcal{F}(L, \mathbf{p}) : \sum_{\ell=0}^{L} \|\mathbf{W}_\ell\|_0 + \|\mathbf{u}_\ell\|_0 \leq s, \ \max_{\ell=0,\ldots,L} \|\mathbf{W}_\ell\|_\infty \vee \|\mathbf{u}_\ell\|_\infty \leq 1, \\ \max_{\ell=0,\ldots,L} \|\mathbf{W}_\ell\|_2 \leq \gamma \ \|f\|_\infty \leq C\} \tag{4.5}$$

This new function space that controls the L_2 norm will contribute to the local strong convexity(details will be in the proof) and we need the estimator $\hat{f} \in \mathcal{F}(L, \mathbf{p}, s, \gamma)$. Once we get the nuclear norm penalized estimator $\hat{f} = \hat{f}_{\hat{\theta}}$ with parameter $\hat{\theta}$, we start from this estimation of model function and boost it with additional neural network $\hat{f}_k \in \mathcal{F}(L, \mathbf{p}, s), k = 1, 2, \ldots, n$. Then we are going to minimize $\mathcal{L}_{\theta_k} = \sum_j \|Y_j - \hat{f}_k(X_j) - \hat{f}(X_j)\|_2$ with respect to their training

parameters θ_k and clip them with range $(-M, M)$ which means $\hat{f}_k(X_i)_{clipped}\mathbb{I}(\|\hat{f}_k(X_i)\|_\infty) = Msgn(\hat{f}_k(X_i))$. In the following sections, we will use $\hat{f}_k$ to represent the clipped or truncated minimizer for simplicity.

The selection of clipped threshold M is also very tricky. Actually when we have the trained network $\hat{f}$, we can plot the distribution of $Y_i - \hat{f}(X_i)$ and find the truncated or clipped point at quantile 0.99. The theorem 8 will guarantee the existence of such a clipped point.

Algorithm 5 Truncated network step

System Initialization
Input data X_i and output data Y_i $i = 1, 2, \ldots,$,number of epochs N and number of points T,batch size B, truncated neural network function $\hat{f}_k$ with trainable parameter $\theta_k, k = 1, 2, \ldots n$ and clipping threshold M. Trained nuclear norm penalized neural network $\hat{f}$
for *epoch* $= 1, 2, \ldots, N$ **do**
  for $j = 1, 2, \ldots, T$ **do**
    for $k = 1, 2, \ldots n$ **do**
      Compute the loss:

$$\mathcal{L}_{\theta_k} = \|Y_j - \hat{f}_k(X_j) - \hat{f}(X_j)\|^2$$

      while θ_k not converged **do**
        Update parameters: $\theta_k \leftarrow \text{ADAM}(\mathcal{L}_{\theta_k})$
      end while
    end for
  end for
end for
Clip the truncated minimizer f_k by range (-M,M)
Output trained truncated network function $\hat{f}_k, k = 1, 2, \ldots, n$ with parameters $\hat{\theta}_k$

4.4 Theorem

In this section, we will discuss the asymptotic analysis of the estimator. First, the nuclear norm penalization ensures the tail bound of the estimator, then the truncated estimator will fill the gap between the true value Y_i and previously estimated model function value $\hat{f}(X_i)$. Before that, we need to define the truncated loss function and gather some assumptions needed.

Define truncated neural network $f_1, f_2, ..., f_n$ clipped at (-M,M), and truncated loss function $\mathcal{L}_{truncated} = x^2\mathbb{I}(\|x\| < M) + M^2\mathbb{I}(\|x\| > M)$, then $\psi(x) = \nabla\mathcal{L}_{truncated} = x\mathbb{I}(\|x\| < M)$

Before we discuss asymptotic analysis theorems, we will need some assumptions.

Assumption 4 (A(1))**.** The true regression function $f \in \mathcal{G}(q, \mathbf{d}, \mathbf{t}, \boldsymbol{\beta}, \mathbf{K})$

This assumption restricts the space of model function to be measurable and composition of several functions $f_0 = g_q \circ g_{q-1} \circ ... \circ g_1 \circ g_0$ where each component function g_i are β-Hölder function.

Assumption 5 (A(2))**.** The truncated estimator $\hat{f}_k \in \mathcal{F}(L, \mathbf{p}, s)$, where $L \asymp \log(T^\nu)$, $s \asymp (T^\nu)^{1/(\theta+1)}$, and $\min_{l=1,...,L} p_l \asymp (T^\nu)^{1/(\theta+1)}$, for $\theta = \min_{\ell=0,...,q} \frac{2\beta_\ell^*}{t_\ell}$ and $\nu \geq 0$.

This assumption restricts the function space of the neural network. The number of layers and the number od hidden neurons grow as the dataset size grows.

Assumption 6 (A(3))**.** The loss function $\mathcal{L}_{truncated}(\cdot)$ is an absolutely continuous convex function on $\mathbb{R}^q$ with derivative $\psi(\cdot) := \nabla\mathcal{L}_{truncated}(\cdot)$ existing almost everywhere.

Assumption 7 (A(4))**.** There exist finite constants κ and c_1 such that for all $x \in \mathbb{R}^q$ and $\|x'\| < \kappa$,

$$\|\psi(x + x') - \psi(x)\| < c_1.$$

This assumption ensures the absolute continuity of the derivative of the loss function.

Assumption 8 (A(5))**.** There exists a finite constant c_2 such that

$$\sup_{j \leq T} \mathbb{E}\|\psi(Q_j + u) - \psi(Q_j)\|^2 < c_2\|u\|, \quad \text{as } \|u\| \to 0.$$

This assumption ensures the Lipchitz condition of the expectation of the derivative of the loss function on the error distribution.

Assumption 9 (A(6))**.** $\sup_{j\leq T}\mathbb{E}\{\psi(Q_j)^2\} = \mathcal{O}(T^{-\nu})$ for some constant $\nu \geq 0$, and $\mathbb{E}\{\psi(Q_j)\} = 0$. There exist finite constants vector $\delta_j \in \mathbb{R}^q$, $j = 1, \ldots, T$ such that

$$0 < \inf_{j\leq T} \delta_j \leq \sup_{j\leq T} \delta_j < \infty$$

and

$$\sup_{j\leq T} |\mathbb{E}\{\psi(Q_j + u)\} - \delta_j u| = o(u), \quad \text{as } \|u\| \to 0.$$

This assumption restricts the turbulence of the loss function derivative in the middle part of the error distribution, which should be controlled by the number of data points.

Assumption 10 (A(7))**.** The input data in dataset $\mathcal{D}_T = \{(X_i, Y_i)\}$ X_i follow the distribution $\mathcal{P}$ where for any given ϵ, there exists a δ, such that $P_{X\sim\mathcal{P}}(\|X\| \geq \delta) \leq \epsilon$

This assumption is also important because it restricts the distribution of input data with some regularization.

4.4.1 Low rank approximation

In this section, we will discuss the two theorems ensuring the existence and robustness of the nuclear norm penalization minimizer and the tail-bound property.

Theorem 7. *Given the model function $f \in \mathcal{G}(q, \mathbf{d}, \mathbf{t}, \boldsymbol{\beta}, \mathbf{K})$ (Assumption A(1)) and the nuclear norm penalization over each layer in feed forward neural network $\hat{f} \in \mathcal{F}(L, \mathbf{p}, s, \gamma)$ without bias vectors, we will get a robust minimizer $\hat{f}_{robust} \in \mathcal{F}(L, \mathbf{p}, s, \gamma)$ close enough to the true minimizer $f^* = \arg\min_{\hat{f}} \mathcal{L}(\hat{f}, f)$ given that the input data distribution is not heavy tailed (Assumption A(7))and γ sufficiently small, then for any δ and any given $X_i \in \mathcal{D}_T$*

$$P(\|f^*(X_i) - \hat{f}_{robust}(X_i)\| > \delta) \to 0 \text{ uniformly as } T \to \infty \tag{4.6}$$

Theorem 8. *Given the model function $f \in \mathcal{G}(q, \mathbf{d}, \mathbf{t}, \boldsymbol{\beta}, \mathbf{K})$(Assumption A(1)) and the nuclear norm penalization over each layer in a feed forward neural network space $\mathcal{F}(L, \mathbf{p}, s, \gamma)$, for*

any $\epsilon > 0$, there exists an constant M and with the global minimizer $\hat{f} \in \mathcal{F}(L, \mathbf{p}, s.\gamma)$ we have

$$\mathbb{E}_{X_i \sim P}(\|Y_i - \hat{f}(X_i)\| > M) \leq \epsilon \tag{4.7}$$

And then for the robust minimizer $\hat{f}_{robust} \in \mathcal{F}(L, \mathbf{p}, s, \gamma)$ we have

$$P(\|Y_i - \hat{f}_{robust}(X_i)\| > M) \to 0 \text{ uniformly as } T \to \infty \tag{4.8}$$

The next theorem gave an asymptotic analysis of the truncated network estimator $\hat{f}_k \in \mathcal{F}(L, \mathbf{p}, s)$ and the mean estimator $\frac{1}{n}\sum_{k=1}^{n} \hat{f}_k$.

Theorem 9. *Given the trained nuclear norm penalized network function $\hat{f}$ and truncated network function $\hat{f}_k \in \mathcal{F}(L, \mathbf{p}, s)$ clipped in range (-M,M), with assumptions A(1)-A(7), then for any given δ*

$$P(\mathbb{E}_{X_i \sim P}(\|Y_i - \hat{f}(X_i) - \hat{f}_k(X_i)\| > \delta)) \to 0 \text{ as } T \to \infty \tag{4.9}$$

$$P(\mathbb{E}_{X_i \sim P}(\|Y_i - \hat{f}(X_i) - \frac{1}{n}\sum_{k=1}^{n} \hat{f}_k(X_i)\| > \delta)) \to 0 \text{ as } T \to \infty \tag{4.10}$$

where $Y_i = f(X_i) + Q_i$

Remark The theorem 9 provides asymptotic analysis of both a single truncated neural network estimator and the mean of multiple truncated neural network estimators.

4.5 Experiment setting

In this section, we will use different types of contamination mechanisms and apply our two-step algorithm. We will compare our algorithm with feed-forward network with regularization and normal feed-forward network. In experiments on synthetic data, we use predefined

feed-forward network with one hidden layer as the model network function f. We define $f(x) = W_2(\sigma(W_1 x))$ where $W_1 \in \mathbb{R}^{100\times 20}, W_2 \in \mathbb{R}^{20\times 20}, x \in R^{100}$ then $f : \mathbb{R}^{100} \to \mathbb{R}^{20}$. We also define dataset $\mathcal{D} = \{X_i, Y_i\}, i = 1, 2, 3...., 10000$ and $Y_i = f(X_i), X_i \in \mathbb{R}^{100}, Y_i \in \mathbb{R}^{20}$. Each element in X_i, W_1, W_2 are sampled from the standard normal distribution. Here, $d_{in} = 100, d_{out} = 20$. In the experiments, we use 80% of the total data points(8000 points) as training data and used the rest 20%(2000 points) as testing data to calculate the test loss.

Case 1: ***Cauchy contamination*** Here we use the noise following Cauchy distribution to contaminate the output.

Set the $\epsilon = 0.01$,each entry of $Q_i \sim \epsilon Cauchy + (1-\epsilon)N(0,1)$

Case 2: ***Random Flip*** Here we apply a frequently used contamination mechanism. We randomly flip the output value with a given rate.

Set the $\epsilon = 0.02$, $Q_i = -2f(X_i)$ with probability ϵ and $Q_i = 0$ with probability $1-\epsilon$

Case 3: ***Input noise*** In most of the experimental settings, the noise or the contamination is added to the output. Here, we use the contamination added to the input. We add random noise to each entry of the selected input data with random selection probability ϵ.

Set $\epsilon = 0.2$, $Q_i = f(X_i + \epsilon_{X_i}) - f(X_i)$ where $\epsilon_{X_i} \in \mathbb{R}^{100}$ and each entry of$\epsilon_{X_i} \sim N(0, 0.2^2)$ with probability ϵ and $Q_i = 0$ with probability $1-\epsilon$.

Case 4: ***Input noise& Random Flip*** Here we use both the contamination added to the input and random flip of the output value with a given rate.

Set $\epsilon_1 = 0.02, \epsilon_2 = 0.2$, $Q_i = -f(X_i+\epsilon_{X_i}) - f(X_i)$ where $\epsilon_{X_i} \sim N(0, 0.2^2)$ with probability $\epsilon_1\epsilon_2$, $Q_i = f(X_i+\epsilon_{X_i}) - f(X_i)$ where $\epsilon_{X_i} \sim N(0, 0.2^2)$ with probability $(1-\epsilon_1)\epsilon_2$, $Q_i = -2f(X_i)$ with probability $(1-\epsilon_2)\epsilon_1$,$Q_i = 0$ where $\epsilon_{X_i} \sim N(0, 0.2^2)$ with probability $(1-\epsilon_1)(1-\epsilon_2)$

Case 5: ***Input noise& Cauchy contamination*** Here we use both the contamination added to the input and the Cauchy distribution contamination.

Set $\epsilon_1 = 0.2, \epsilon_2 = 0.01$, $Q_i = f(X_i+\epsilon_{X_i}) - f(X_i) + \epsilon_2 Cauchy + (1-\epsilon_2)N(0,1)$ where $\epsilon_{X_i} \in \mathbb{R}^{100}$ and each entry of$\epsilon_{X_i} \sim N(0, 0.2^2)$ with probability ϵ_1, $Q_i = \epsilon_2 Cauchy + (1-\epsilon_2)N(0,1)$ with probability $(1-\epsilon_1)$

The results are presented in the table 4.1 Here in the table header, **MLP** is the feed-forward network with 2 hidden layers (Baseline) **MLP+T** is the same MLP + Truncation step, **MLP+R** is the same MLP with L_1, L_2 regularization and dropout, **MLP+R+T** is the same regularized MLP+R plus Truncation step, **Low Rank** is the Output of low rank step (step 1), **Two step** is Our proposed algorithm.

	MLP	**MLP+T**	**MLP+R**	**MLP+R+T**	**Low Rank**	**Two Step**
Input noise	100(0)	99.4(0.2)	101.2(0.2)	100.4(0.3)	99.6(0.6)	**98.8(0.6)**
Input noise&Flip	100(0)	99.9(0.1)	100.2(0.4)	100.0(0.4)	99.4(0.3)	**99.3(0.4)**
Input noise &Cauchy Contamination	100(0)	98.9(2.7)	102.72(14.0)	99.4(13.5)	96.0(6.7)	**91.0(7.5)**
Flip	100(0)	99.8(0.1)	100.3(0.4)	100.0(0.3)	99.6(0.5)	**99.3(0.4)**
Cauchy Contamination	100(0)	99.8(2.2)	112.0(9.1)	104.4(4.1)	100.8(1.6)	**97.4(0.4)**

Table 4.1: Benchmark with different algorithms and contamination scenarios

We also test our algorithm on real datasets. One of the datasets we benchmark is Bike share data [32], which includes the weather and seasonal conditions as features and the total number of bikes in the system as the target. Similarly, we use the test loss in MLP setting as the baseline and we divide the test loss in other settings by this baseline results and present them in percentage. The results can be found in table 4.2

Here since the target is the total number of bikes, which should always be positive, so it is unrealistic to flip the target. The rest of the contamination is included. In the input noise scenario, we set the noise added to variable X is set to a normal distribution $N(0, 0.16^2)$. In the input noise combined with Cauchy Contamination scenario, we set the input noise as $N(0, 0.12^2)$ combined with $0.9Cauchy$ contamination on the target. In the Cauchy contamination-only scenario, we set the contamination to be $0.9Cauchy$ as well.

	MLP	**MLP+T**	**MLP+R**	**MLP+R+T**	**Low Rank**	**Two Step**
Input noise	100	97.2	87.5	85.5	86.0	**85.0**
Input noise &Cauchy Contamination	100	98.5	100.4	99.0	98.5	**97.9**
Cauchy Contamination	100	98.9	105.0	104.2	97.5	**97.3**

Table 4.2: Benchmark with different algorithms and different contamination scenarios in Bike Sharing data

Besides the regression task, we also test our algorithm in the classification task. Here we

use the Cifar-10 data [53] and set the input noise scale to 0.3. Because we want to include enough noise, but too much noise might make the network learn nothing.

	MLP	**MLP+T**	**MLP+R**	**MLP+R+T**	**Low Rank**	**Two Step**
Input noise	36.4	40.1	35.8	41.2	36.2	**41.5**

Table 4.3: Benchmark with different algorithms and different contamination scenarios in Cifar-10 data

We can also add some convolutional layers before the MLP to increase the accuracy for classification tasks. Here the table 4.4 demonstrates the performance benchmark when we add the convolutional blocks.

The low rank neural network we are using here is defined as $\hat{f}(x) = W_3\sigma(W_2(\sigma(W_1 Conv(x))))$ where $W_1 \in \mathbb{R}^{2048\times512}, W_2 \in \mathbb{R}^{512\times512}, W_3 \in \mathbb{R}^{512\times10}, x \in \mathbb{R}^{32\times32\times3}, Conv : \mathbb{R}^{32\times32\times3} \to \mathbb{R}^{2048}$ then $f : \mathbb{R}^{100} \to \mathbb{R}^{20}$. And the penalization is put on the nuclear norm of $F_1 = W_1 Conv(x)), F_2 = W_2(\sigma(W_1 Conv(x))), F_3 = W_3\sigma(W_2(\sigma(W_1 Conv(x))))$

	MLP	**MLP+T**	**MLP+R**	**MLP+R+T**	**Low Rank**	**Two Step**
Input noise	57.6	58.1	54.0	58.0	57.6	**58.5**

Table 4.4: Benchmark with different algorithms and different contamination scenarios in Cifar-10 data with CNN layers

4.6 Supplements

4.6.1 Experiments setting

We use random seeds (randomly selected) 2,42,152,362,572 for each setting and get 5 results for each setting, and calculate the mean and standard deviation. Since the different contamination with each different seed may result in completely different error expectation, we set the MLP as the baseline, and divide the test error of other experimental settings by the MLP test error and average across the 5 different seeds to get the mean and standard deviation.

4.7 Proofs

4.7.1 Proof of Theorem 7

In this paper, the loss function of the first step is defined as either L_2 loss or cross-entropy loss with nuclear norm penalization. We assume that the population risk is convex to ensure tractable minimization. Moreover, in order to ensure identifiability of the parameter θ^*, we impose two standard regularity conditions from paper [73] on the population risk. These properties are defined in terms of the error of the first-order Taylor approximation of the population risk, i.e. defining,

$$\tau(\theta_1, \theta_2) := \mathcal{R}(\theta_1) - \mathcal{R}(\theta_2) - \langle \nabla \mathcal{R}(\theta_2), \theta_1 - \theta_2 \rangle,$$

where $\mathcal{R}(\theta) = \mathbb{E}_X[\mathcal{L}(\theta, X)]$

we assume that

$$\frac{\tau_\ell}{2}\|\theta_1 - \theta_2\|_2^2 \leq \tau(\theta_1, \theta_2) \leq \frac{\tau_u}{2}\|\theta_1 - \theta_2\|_2^2. \tag{4.11}$$

The upper bound is easy to get given $\hat{f} \in \mathcal{F}(L, \mathbf{p}, s)$, then Hessian matrix of $\mathcal{R}(\theta)$ $H(\mathcal{R}) \simeq \mathbb{E}[H(\mathcal{L})] = \mathbb{E}[J^T J]$ where $J = \frac{\partial \hat{f}}{\partial \theta} = (\frac{\partial \hat{f}}{\partial W_L}, \frac{\partial \hat{f}}{\partial W_{L-1}}, \dots \frac{\partial \hat{f}}{\partial W_1})$ By Gaussian-Newton approximation. Because each element in W_i matrix is bounded by a universal constant C, the Jacobi matrix J is then bounded, given that the input data is bounded and so is the Hessian matrix. Then by Taylor expansion, the upper bound holds.

Let $\mathcal{L}(\theta) = L(\theta) + \frac{\lambda_2}{2}\|\theta\|_2^2 + \lambda_* \sum_{l=0}^{L} \|F_l\|_*$ where L_θ is the L_2 loss. Minimization of $\|F_l\|_*$ is equivalent to minimization of the $\|W_l\|_*, l = 1, 2, \dots, L$. Also, since the neural network $\hat{f} \in \mathcal{F}(L, \mathbf{p}, s, \gamma)$, it is equivalent to penalize the L_2 norm of parameter θ

When θ is sufficiently small given by the constraints $\|\theta\| \leq \delta_\theta$. Then, in the ball $U = \{\theta : \|\theta\| \leq \delta_\theta\}$,neural network $\hat{f}(x; \theta) \simeq A(\theta)x + b(\theta)$ becomes approximately affline which is strong convex so that we have $\mathcal{L}(\theta)$ is μ-strongly convex over U for some $\mu > 0$.

That is,

$$\nabla^2 \mathcal{L}(\theta') \succeq \mu I \quad \text{for all} \quad \theta' \in U.$$

Thus, we can claim the strong convexity of loss function.

Here we introduce the gradient estimator and the resulting contraction parameter.

Definition 1. A function $g(\theta; \mathcal{D}_n, \delta)$ is a gradient estimator, if for functions α and β, with probability at least $1 - \delta$, at any fixed $\theta \in \Theta$, the estimator satisfies the following inequality:

$$\|g(\theta; \mathcal{D}_T, \delta) - \nabla \mathcal{R}(\theta)\|_2 \leq \alpha(T, \delta)\|\theta - \theta^*\|_2 + \beta(n, \delta). \tag{4.12}$$

Definition 2 (Stability)**.** *A gradient estimator is stable for a given risk function $\mathcal{R} : \Theta \mapsto \mathbb{R}$ if for some $\phi \in [0, \tau_\ell)$,*

$$\alpha(B, \tilde{\delta}) < \tau_\ell - \phi.$$

We denote by κ the following contraction parameter:

$$\kappa := \sqrt{1 - \frac{2\eta\tau_\ell\tau_u}{\tau_\ell + \tau_u}} + \eta\alpha(B, \tilde{\delta}),$$

and note that $\kappa < 1$. With these definitions in place we state our main result on the stability of gradient descent:

Lemma 3. *Suppose that the gradient estimator satisfies the condition in (11) and is stable for the risk function $\mathcal{R} : \Theta \mapsto \mathbb{R}$. Then Algorithm 1 initialized at θ^0 with step-size $\eta = \frac{2}{\tau_\ell + \tau_u}$, returns iterates $\{\hat{\theta}^t\}_{t=1}^T$ such that with probability at least $1 - \delta$ for the contraction parameter κ above we have that,*

$$\|\hat{\theta}^t - \theta^*\|_2 \leq \kappa^t \|\theta^0 - \theta^*\|_2 + \frac{1}{1 - \kappa}\beta(B, \tilde{\delta}).$$

The first term is decreasing in T, while the second term is increasing in T. This suggests that for a given B and δ, we need to run just enough iterations for the first term to be

bounded by the second. Hence, we can fix the number of iterations N^* as the smallest positive integer such that:

$$N \geq \log_{1/\kappa}\left(\frac{(1-\kappa)\|\theta^0-\theta^*\|_2}{\beta(B,\tilde{\delta})}\right).$$

Then $\theta^t \to \theta^*$

To prove the theorem 7, firstly, given a sufficiently large batch size B, we have

$$\|g(\theta;\mathcal{D}_{(j-1)B:jB},\delta)-\nabla\mathcal{R}(\theta)\|_2 = \|\frac{1}{B}\sum_{i=0}^{B-1}\nabla\mathcal{L}(\theta,X_{(j-1)B+i},Y_{(j-1)B+i})-\mathbb{E}_X[\nabla\mathcal{L}]\|_2 \leq \frac{C}{B} \quad (4.13)$$

Then we have $\alpha(B,\tilde{\delta})=0, \beta(B,\tilde{\delta})=\frac{1}{B}$, and we have $\|\hat{\theta}^t-\theta^*\|_2 \leq \kappa^t\|\theta_0-\theta\|_2 \to 0$

Once we have the convergence of the parameters $\hat{\theta}^t \to \theta^*$. We can claim the network $\hat{f}$ converged to true minimizer $\hat{f}^*$.

4.7.2 Proof of Theorem 8

Given the network with nuclear penalization for each layer:

$$\hat{f}(x) = W_1\sigma(W_2\sigma(...W_Lx)) \quad (4.14)$$

Define $M = \max_j ||\hat{f}(X_{(j-1)B:jB})||_*$,if $\exists X_i$, such that, $||\hat{f}(X_i)||_2 > M$, then $||\hat{f}(X_i)||_2^2 = \langle\hat{f}(X_i),\hat{f}(X_i)\rangle \leq ||\hat{f}(X_{(j-1)B:jB})||_* * \hat{f}(X_i) = M^2$, a contradiction Also, $\exists M^*$ such that $\mathbb{E}[||Y_i||_\infty > M^*] = \epsilon$, thus finished the proof

4.7.3 Truncated network

Define network $f_1, f_2, ..., f_n$ clipped at (-M,M), We will firstly prove this multivariate version of lemma in paper [93]

Lemma 4. *Under Assumptions* (A1)–(A6)*, we have*

$$\frac{1}{T}\sum_{i=1}^{d}\sum_{j=1}^{T}(\hat{f}^i(X_j)-f^i(X_j))^2 = O_p\left((T^\nu)^{-\frac{\theta}{\theta+1}}\log^6(T^\nu)\right),$$

where $\nu \geq 0$*, and*

$$\theta = \min_{\ell=0,\ldots,q}\frac{2\beta_\ell^*}{t_\ell}.$$

Proof of lemma 4

Given network $\hat{f} = (\hat{f}^1, \hat{f}^2, \ldots, \hat{f}^q)$ where each output entry $\hat{f}^i$ is a small network with output dimension 1 and model fuction$f = (f^1, f^2, \ldots, f^d)$ where each entry $f^i \in \mathcal{G}(q, \mathbf{d}, \mathbf{t}, \boldsymbol{\beta}, \mathbf{K})$ is a one dimension function. Then for each small network, we define loss function $\mathcal{L}_i = \mathcal{L}_{truncated}(\mathbf{x}) = \mathcal{L}_{truncated}(x_i)$, then $\mathcal{L}_i' = x_i\mathbb{I}(\|\mathbf{x}\| \leq M)$ We need to prove the A(1)-A(6) assumptions hold for all entries. Here the u_i, δ_{ij}, x_i' represents the ith entry of vector u, δ_j, x'.

(A1) The true regression function $f \in \mathcal{G}(q, \mathbf{d}, \mathbf{t}, \boldsymbol{\beta}, \mathbf{K})$.

(A2) The truncated estimator $\hat{f} \in \mathcal{F}(L, \mathbf{p}, s)$, where $L \asymp \log(T^\nu)$, $s \asymp (T^\nu)^{1/(\theta+1)}$, and $\min_{l=1,\ldots,L} p_l \asymp (T^\nu)^{1/(\theta+1)}$, for $\theta = \min_{\ell=0,\ldots,q}\frac{2\beta_\ell^*}{t_\ell}$ and $\nu \geq 0$.

(A3) The loss function $\mathcal{L}_i(\cdot)$ is an absolutely continuous convex function on $\mathbb{R}$ with derivative $\psi(\cdot) := \mathcal{L}_i'(\cdot)$ existing almost everywhere.

(A4) There exist finite constants κ and c_1 such that for all $x \in \mathbb{R}$ and $|x_i'| < \kappa$,

$$|\psi(x + x_i') - \psi(x)| < c_1.$$

(A5) There exists a finite constant c_2 such that

$$\sup_{j\leq T}\mathbb{E}|\psi(Q_j + u_i) - \psi(Q_j)|^2 < c_2|u_i|, \quad \text{as } |u_i| \to 0.$$

(A6) $\sup_{j\leq T}\mathbb{E}\{\psi(Q_j)^2\} = \mathcal{O}(T^{-\nu})$ for some constant $\nu \geq 0$, and $\mathbb{E}\{\psi(Q_j)\} = 0$. There exist finite constants δ_j, $j = 1, \ldots, T$ such that

$$0 < \inf_{j\leq T} \delta_{ij} \leq \sup_{j\leq T} \delta_{ij} < \infty$$

and

$$\sup_{j\leq T} |\mathbb{E}\{\psi(Q_j + u_i)\} - \delta_j u_i| = o(u_i), \quad \text{as } |u_i| \to 0.$$

Then we can check each assumption again. A(1) is satisfied automatically so as is assumption A(2). By the definition of multivariate convex function A(3) is also satisfied. For assumption A(4) and A(5), similarly, for each i, we set only ith entry in vector x, x', namely x_i, x_i' as nonzero. Set only ith entry of u as non zero. Then by the assumptions, the A(4) and A(5) are satisfied. For Assumption A(6), We can just set δ_j to be vector with only ith entry positive.

Then by the lemma

$$\frac{1}{T}\sum_{i=1}^{T}(Y_{ij} - \hat{f}^i(X_j))^2 \leq O_p\left((T^\nu)^{-\frac{\theta}{\theta+1}}\log^6(T^\nu)\right)$$

$$\frac{1}{T}\sum_{j=1}^{T}\|Y_j - \hat{f}(X_j)\|^2 = \frac{1}{T}\sum_{j=1}^{d}\sum_{i=1}^{T}(Y_{ij} - \hat{f}^i(X_j))^2 \leq O_p\left((T^\nu)^{-\frac{\theta}{\theta+1}}\log^6(T^\nu)\right)$$

Assume for each minimizer f_k, there exists a corresponding minimizer $\tilde{f}_k$ where $f_k(X_i)\mathbb{I}(Y_i - \hat{f}(X_i) \leq M) = \tilde{f}_k(X_i)\mathbb{I}(Y_i - \hat{f}(X_i) \leq M)$ under loss $\mathcal{L}$.

$$\sum_{i=1}^{T}(Y_i-\hat{f}(X_i)-f_k(X_i))^2=\sum_{i=1}^{T}(Y_i-\hat{f}(X_i)-f_k(X_i))^2\mathbb{I}(Y_i-\hat{f}(X_i)\leq M)$$
$$+(Y_i-\hat{f}(X_i)-f_k(X_i))^2\mathbb{I}(Y_i-\hat{f}(X_i)>M)$$
$$=\sum_{i=1}^{T}(Y_i-\hat{f}(X_i)-\tilde{f}_k(X_i))^2\mathbb{I}(Y_i-\hat{f}(X_i)\leq M)+(Y_i-\hat{f}(X_i)-f_k(X_i))^2\mathbb{I}(Y_i-\hat{f}(X_i)>M)$$
$$=\sum_{i=0}^{T}(Y_i-\hat{f}(X_i)-f_k(X_i))^2\mathbb{I}(Y_i-\hat{f}(X_i)>M)+\mathcal{L}(Y_i-\hat{f}(X_i),\hat{f}_k(X_i))$$
$$=\mathcal{L}(Y_i-\hat{f}(X_i),\hat{f}_k(X_i))+O(M)\epsilon \quad (4.15)$$

As a result, when we have $\hat{f}_k=\arg\min_{\hat{f}_k}\sum_{i=1}^{T}(Y_i-\hat{f}(X_i)-f_k(X_i))^2$, $\hat{f}_k=\tilde{f}$ on $\mathbb{I}(Y_i-\hat{f}(X_i)\leq M)$, $\hat{f}_k$ minimize the loss $\mathcal{L}$. Then we have $\frac{1}{T}\sum_{i=1}^{T}\|\hat{f}+\hat{f}_k-f\|_2=O_p\left((T^\nu)^{-\frac{\theta}{\theta+1}}\log^6(T^\nu)\right)$ and $\frac{1}{T}\sum_{i=1}^{T}\|\hat{f}+\frac{1}{n}\sum_{k=1}^{n}\hat{f}_k-f\|_2=O_p\left((T^\nu)^{-\frac{\theta}{\theta+1}}\log^6(T^\nu)\right)$.

Then, by the chebyshev's inequity, $\frac{1}{T}\sum_{i=1}^{T}\|\hat{f}+\hat{f}_k-f\|_2=O_p(1)+\mathbb{E}_{X_i\sim P}[(Y_i-\hat{f}(X_i)-\hat{f}_k(X_i))^2]$, as a result, $P(\mathbb{E}_{X_i\sim P}(|Y_i-\hat{f}(X_i)-\frac{1}{n}\sum_{k=1}^{n}\hat{f}_k(X_i)|>\delta))\to 0$ as $T\to\infty$, similarly, we can prove this also hold for mean estimator $\frac{1}{n}\sum_{k=1}^{n}\hat{f}_k$

Chapter 5

ARReST:Antagonistic Redundancy Reduction in Patch Selection of Whole Slide Images for Efficient Indexing and Retrieval in Histopathology

5.1 Introduction

In this chapter, I will discuss a project collaborating with Scientists at Mayo Clinic.

With the increasing number of whole slide images (WSIs) in digital pathology and the growing need for efficient search and retrieval of WSI archives, effective indexing has become a critical necessity [87]. Currently, all unsupervised patch selection methods, such as Yottixel's mosaic [50], SPLICE [3], and SDM [82], employ a "divide" approach to manage the gigapixel scale of each WSI. These methods break down WSIs into a small, representative set of patches that facilitate indexing and retrieval. However, significant bottlenecks and challenges remain in this field. An effective WSI search engine and indexing strategy must account for storage requirements, cost efficiency, and the ability to perform fast comparisons during both the

indexing and search processes.

Storage costs and management are critical concerns in digital pathology [7, 42]. Several studies have highlighted the significant expenses associated with handling and storing WSIs [6, 43]. The size of each WSI varies depending on magnification and resolution [11, 27], typically ranging from 5 to 30 gigabytes per image. With the continuous digitization of new WSIs, it is estimated that storing them in an indexed archive, referred to as the Atlas, could require 50–150 petabytes annually [8], making it highly inefficient.

Existing unsupervised methods, including Yottixel's mosaic [50], SPLICE [3], and SDM [82], extract a representative set of patches from each WSI, reducing indexing storage to approximately 5–10% of the original WSI size. While this is a significant improvement, the resulting storage demand—estimated at 5–15 petabytes—remains prohibitively large. Given that the cost per petabyte varies across institutions, the financial burden on hospitals can be substantial. Therefore, any further reduction in indexing storage would be highly valuable, facilitating the broader adoption of digital pathology and efficient retrieval solutions.

In the meantime, the comparison of embeddings is also a problem for most of the automated search systems. They usually employ the calculation of the distances between embeddings from incoming WSIs and embeddings in the repository which often includes exhaustive pair-wise calculation and can be very time-consuming. For example, if the incoming WSI is divided and passed through the foundation model and obtains m embeddings, there would be $m \times n$ comparisons for each pair of embeddings when compared with another WSI with n embeddings. Then, each amount of the reduction in the embeddings can result in n less comparisons.

The main contribution of this paper is that we proposed a similarity matching procedure in the search engine which include the removal of redundancy by finding the similar patch embedding pairs from two WSIs with different classes and creating the redundancy repository used as a reference removing the redundancy from the patch embeddings of incoming WSIs. This new approach not only decreases the storage requirement but also accelerates the

comparison process in retrieval stage. The rest of the paper will be organized like this: Section 5.2 will give a brief introduction of current search, retrieval, indexing system, section 5.3 will introduce the proposed similarity matching method in detail and Section 5.4 will show the results evaluated in public TCGA datasets.

5.2 Background knowledge

One of the main issue in digitalized histopathology is the lack of efficient approach to extract information from and deal with the large size WSIs. The current baseline is the search engine proposed in paper [50]. The next few subsections cover the background knowledge of the entire workflow of Patch selection, retrieval and indexing algorithms which applies to many popular algorithms including this baseline Yottixel algorithm. Given the Atlas WSIs with labels, we will break them down into patches and apply the patch selection algorithm to pick a representation set of patches for each WSI and employ further embedding and barcoding methods. Then the incoming new WSIs will go through the same workflow and get predicted labels by leveraging the saved information of Atlas WSIs.

The proposed method in this paper which will be discussed in detail in the next section reduces the storage further by reducing the size of the representation set of patches for each WSI and decreases the searching time after reducing the redundant patches in the incoming WSIs as well.

5.2.1 Tissue segmentation

The first step is preprocessing the WSI, we remove the non-tissue segments like pen marks or white background. We apply $n \times n$ grid boxes in the WSIs and calculate the tissue coverage within each box. We exclude all the boxes with less than 30% of the tissue. We can call each of the remainder of the boxes a patch of this WSI.

5.2.2 Patch selection algorithm

Currently, the most popular approach to extracting information from WSIs is to divide or break the WSI image into patches and only select a small representation sample of each WSI. This step is called patch selection. Many such algorithms have emerged recently.

Yottixel used to be the state-of-art patching algorithm, the main idea of Yottixel is two step clustering. The first step is the Kmeans clustering based on the color histogram, second step is to sample by the spatial information. With a fixed inner cluster sample rate, extract the patch closest to each cluster centroid.

SPLICE or Sequential Patching Lattice for Image Classification and Enquiry used the color histogram to perform a sequential sampling. After each sample, the similar patches are removed from the sampling pool to encourage diversity of the sampling.

5.2.3 Foundation model embedding

After selecting the representation sample of each WSI, we use a pre-trained or fine tuned model, or most of the time, the foundation model, to extract feature embeddings from each patch. This step saves the storage and also makes the further BOB encoding, retrieval and indexing easier and simplifies the similarity score calculation as well.

5.2.4 BOB barcoding

Binary encoding is a common practice in many algorithms, which alleviates the storage burden and increases the retrieval speed. The MinMax Algorithm from paper [86] maps the feature embeddings using the sign of 1d deviation to the binary code which enable fast distance analysis by calculating the Hamming distance instead of the Euclidean distance directly from embeddings.

5.2.5 Top k majority voting prediction

In this step, we retrieve the information from Atlas embeddings and use it to predict the label of incoming WSIs. For each incoming WSI, we apply the previous workflow and got its BOB barcodes. We calculate the distance between it and barcodes of each WSI in Atlas, which is defined as the median minimum Hamming distance of all the barcode pairs between the incoming and selected Atlas WSI. We extract k closest Atlas WSIs barcodes and find their labels. After majority voting, we then predict the label of the incoming WSI.

5.3 Proposed Methods

The main idea behind the method is that if we select two representation patch embeddings from two WSIs from different classes and compare pairwisely, the pair will have more similarity when both patches are from normal tissue compared to those from abnormal or cancer tissue. This motivates us to propose a new methodology for the redundancy removal process. Usually, we don't have the labeled normal tissue as the training repository to be used to identify the normal cases in each cancer class. However, when we have the labeled class for each type of cancer, then we can use the pairwise comparison trick to create a repository and use this repository to remove the redundant normal patches from each WSI.

The first step is to implement similarity matching and create a redundancy repository from this process. Given the labeled WSIs (W_i, L_i) where W_i represents the WSI and L_i represents the label. We use a patch selection algorithm to break down each WSI into representation sample patches $\mathbf{P}_i = \{P_{ij}\}_{j=1,2,\ldots n_i}$ where P_{ij} refers to a patch in the sample of a WSI and n_i is the sample size of WSI W_i.

The figure 5.1 demonstrates the idea of similarity matching. For each WSI, the patch selection algorithm(like Yottixel, Splice, SDM,etc) will be applied to it and create a representation set of patches and then passed through a foundation model to get embeddings $\mathbf{P}_i$. Then we keep selecting two patched and embedded samples from two WSI with different

labels and get a pairwise similarity score. Then, we remove the pair of patched embeddings whose similarity score lower than q quantile of the whole similarity score repository from both patched samples and store it separately as a redundancy repository. For a better understanding of the framework, we will omit the embedding process in the figure and show directly the similar patches(usually both are normal) are removed.

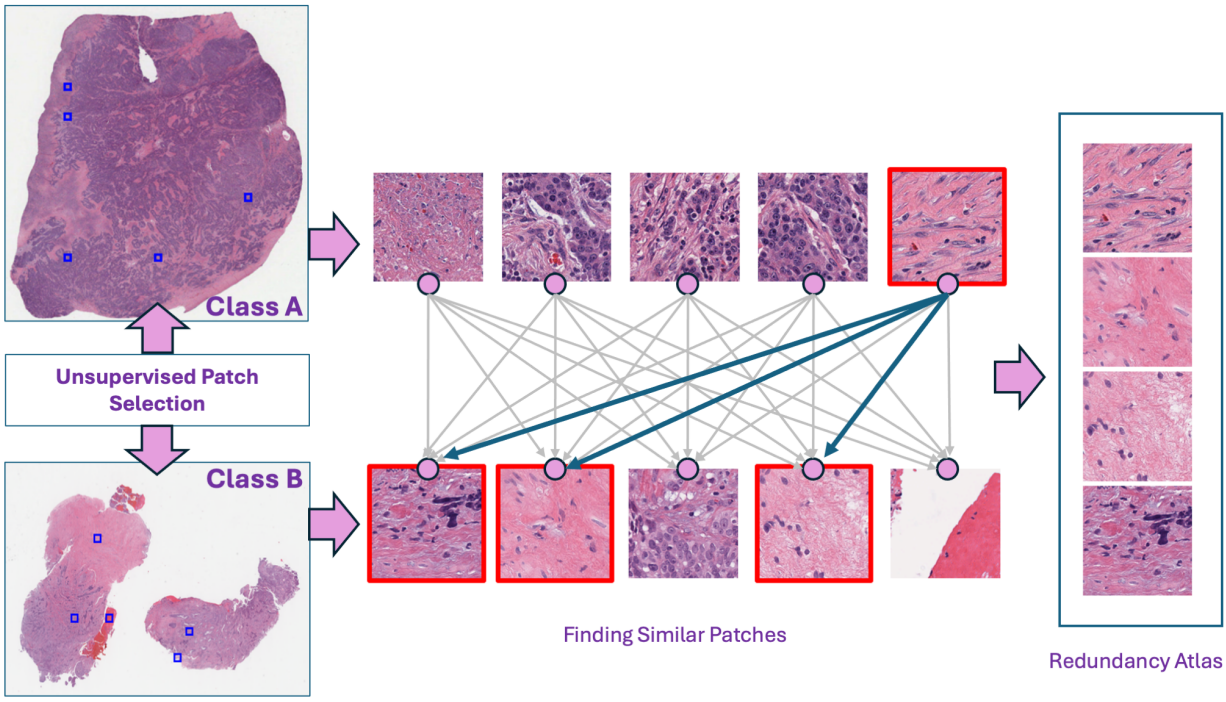


Figure 5.1: Workflow to establish an atlas of redundant patches. Two WSIs from different classes (e.g., two different cancer subtypes) are selected first. an unsupervised patch selection method is applied to select some patches. These patches then are compared to find similar patches (common between class A and B). They are subsequently added to the Redundancy Atlas.

Once we get the redundancy repository R and updated atlas patch embeddings and labels $(\hat{\mathbf{P}}_i, L_i)$, a common downstream approach is to encode these embeddings as BOB Barcodes $\hat{\mathbf{B}}_i$ for further storage saving and more efficient distance calculation as the hamming distance calculation in barcodes are generally faster and more straightforward compared to other types of distance.

The algorithm 7 shows the details of the downstream process and testing procedure. For each incoming WSI W_i, we pass it through a patch selection algorithm to get representation

sample patches and then employ a foundation model mapping pathes to embeddings $\mathbf{P}_i \in \mathcal{P}$ of each WSI. Then for each patch embedding $P_{ik} \in \mathbf{P}_i$, we calculate the similarity score between this embedding and every embedding R_j in the redundancy repository R. If the similarity score is less than the q quantile of Similarity Score repository S_q, we remove this patch embedding from the representation sample. Then we get the updated patch embedding $\tilde{\mathbf{P}}_i$ and its BOB barcode $\tilde{\mathbf{B}}_i$. The next step is to use the atlas BOB Barcodes, labels and patients id $(\hat{\mathbf{B}}_i, L_i, G_i)$ to predict the label with top k majority voting.

The Figure 5.2 demonstrates the whole pipeline of the search engine and indexing method we develop.

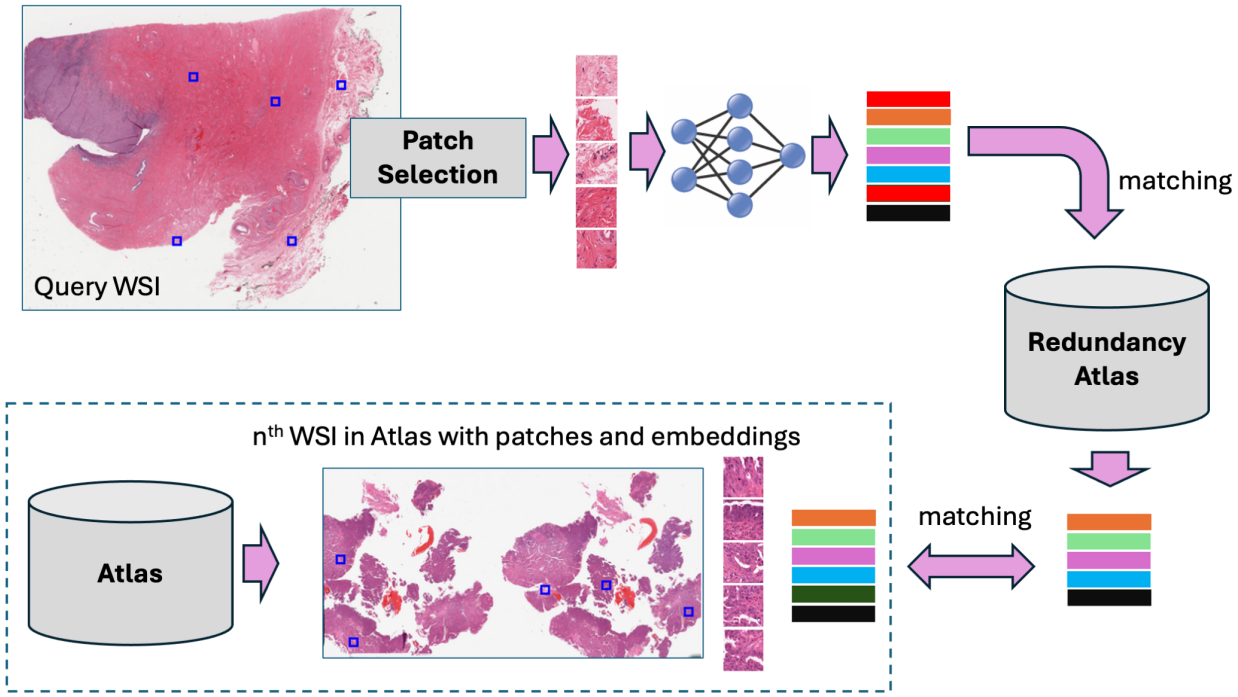


Figure 5.2: The structure of the search engine when an Atlas is used after matching patch embeddings with a Redundancy Atlas eliminates irrelevant patches.

5.4 Results

In this section, we will compare the search capabilities of the extracted embeddings before and after the redundancy removal by similarity matching. We use the Yottixel as the baseline and

patching algorithm, and use 20× magnification and patch size 224 × 224. For the foundation model to extract deep features or embeddings, we use UNI net, which is a pretrained vision encoder for histopathology (100M images, 100K WSIs) developed on internal neoplastic, infectious, inflammatory, and normal tissue and also made publicly available [23].

We use the 5 folds cross validation to evaluate the performance of both circumstances. In each fold, Both embeddings are evaluated by the “leave-patient-out" test. To ensure the comparison between two WSIs from the same patient should not happen, this technique requires all the WSIs from one patient only be in a single fold, and the distribution of labels in each fold are as similar as possible. We implement it by Stratified-Group K fold cross validation technique. Within each fold which will be the test dataset and the rest of the folds will be the atlas. Basically, each WSI in the test dataset will be patched and embedded, and compared with all the embeddings in the redundancy repository, and calculated similarity score. The embeddings have a low similarity score with any of the embeddings in the redundancy repository will be removed. After that, we will get the barcodes of the remaining embeddings and calculate the distance between the barcodes of the embeddings with the barcoded embeddings in the Atlas. Then the top 1,3,5 WSIs will be extracted from Atlas, and we then get the predicted label for the WSI by majority voting.

The data we use is The Cancer Genome Atlas or TCGA [17] with 11,679 WSIs. We subset it by different organs. Table 5.1 shows Accuracy, Macro Average of F1 score, and Weighted Average of F1 score before and after similarity matching in the Top-1, MV3 and MV5 cases.

Compared to the original version of Yottixel, in organs, it shows improvement in accuracy and F1 score. For all the organs in TCGA, it shows the savings in storage. Especially, for some organs like the Adrenal glands and Testicles, the saving is significant (30-60%). This demonstrates the capability and potential that ARReST can save storage in patch selection and retrieval and indexing time while maintaining the same level of retrieval and indexing performance.

Table 5.1: Before redundancy removal versus After redundancy removal.

	Accuracy						Macro Average						
	Top-1		MV@3		MV@5		Top-1		MV@3		MV@5		
Datasets	Before	After	Before	After	Before	After	Before	After	Before	After	Before	After	Saving
Cervix	52	**53**	60	60	61	**62**	**28**	27	**30**	28	25	**28**	24%
Lung	63	**64**	66	66	67	**68**	**23**	22	**24**	21	**22**	20	17%
Testicles	51	**56**	56	**59**	60	60	35	**39**	39	**41**	**44**	42	33%
Colon	86	86	86	**87**	86	86	54	**55**	57	**59**	54	**55**	6%
Esophagus	91	**92**	90	**92**	**91**	90	62	62	62	62	62	62	8%
Eye	**51**	49	**51**	49	**51**	48	**42**	41	**45**	42	**36**	32	5%
Pancreas	**82**	81	87	**88**	88	88	41	**43**	47	**49**	45	**48**	12%
Bladder	**85**	83	87	87	87	**88**	**68**	65	69	69	70	70	3%
Rectum	84	84	85	**86**	87	**88**	62	**64**	57	**59**	53	**55**	15%
Thymus	34	34	**38**	36	**40**	36	27	**28**	**35**	33	**35**	32	12%
Stomach	44	44	49	49	**50**	48	31	**32**	34	34	**36**	34	7%
Skin	85	**88**	90	**91**	91	91	**32**	20	**31**	19	**23**	19	12%
Chest/abdom.	**59**	57	67	67	**70**	69	**34**	31	**36**	34	**36**	34	4%
Thyroid gland	71	**72**	**78**	76	**79**	77	48	48	**49**	43	**48**	43	4%
Head neck	82	**84**	85	85	84	**85**	**45**	37	**39**	25	**29**	25	16%
Soft Tissue	46	46	47	47	46	**47**	44	38	**45**	39	**42**	38	23%
Adrenal glands	79	**81**	79	**82**	80	**81**	**41**	34	**39**	33	**34**	29	60%
Uterus	**75**	74	**76**	75	74	**75**	**53**	50	52	52	**47**	46	9%
Breast	73	**74**	79	**80**	**81**	80	**27**	25	**31**	29	**31**	29	7%
Kidney	**92**	91	93	93	92	92	**79**	76	**78**	76	**72**	69	10%
Brain	63	63	65	65	65	**66**	41	**42**	44	44	44	**45**	13%

5.5 Discussion and Conclusion

In this study, we propose a novel approach, Antagonistic Redundancy Reduction or ARReST aimed at improving the efficiency of indexing and retrieval in whole slide image (WSI) search engines used in digital pathology. By leveraging redundancy reduction among dissimilar classes, our method significantly reduces storage requirements while maintaining the same level of, and in some cases enhancing, retrieval accuracy. The results of our experiments on TCGA datasets demonstrate that ARReST effectively minimizes the number of stored patches without sacrificing the performance of the search engine.

The most significant insight we can get from this approach is that a large portion of normal patches are usually redundant, which appears across multiple WSIs regardless of cancer type. By eliminating these redundant patches early in the process, our method significantly reduces the number of stored patches and maintain the same level of performance.

Beyond storage savings, the proposed approach also speeds up the search process. Since fewer patches need to be compared, retrieval times decrease, making the system more practical for large-scale applications. This is especially important in clinical settings where rapid and accurate retrieval of WSIs can support diagnostic workflows.

However, there are some limitations to our approach. The effectiveness of redundancy

removal depends on the similarity threshold, which might not be optimal for all datasets. Some tissues have greater morphological variation, meaning that an overly aggressive removal process which could risk losing useful information. Future improvements could involve adaptive thresholds that adjust based on dataset characteristics or deep learning models to refine similarity detection.

In summary, ARReST provides a practical and effective way to enhance WSI search engines by reducing redundant storage and speeding up retrieval. By addressing both storage constraints and computational efficiency, this method could help digital pathology scale more effectively as WSI archives continue to grow. The future work could focus on more adaptive and precise similarity matching methods including deep-learning-based similarity measures and the extension to medical imaging outside the field of histopathology.

5.6 Acknowledgments

This project is a joint work with Dr. Hamid R. Tizhoosh and Dr. Saghir Alfasly at Mayo Clinic. I appreciate Dr. Saghir Alfasly for his brilliant idea of similarity matching and insights in this project and the guidance and supervision by Dr. Hamid R. Tizhoosh. I would also like to thank Mayo Clinic for holding me as an intern and giving me the access to its resource and datasets.

Algorithm 6 Create redundancy repository

Require: Patches embeddings $\mathbf{P}_i = \{P_{ij}\}_{j=i,2,...n_i}$, labels L_i,i=1,2,...,n. Similarity Score function f, prespecified quantile q.

Initialize Normal repository $R \leftarrow []$, Similar Score repository $S \leftarrow []$
for $i \leftarrow 1$ to n **do**
  for $j \leftarrow 1$ to n **do**
    if $i \neq j$ **then**
      for $P_{ik} \in \mathbf{P}_i$ **do**
        for $P_{jm} \in \mathbf{P}_j$ **do**
          Append $f(P_{ik}, P_{jm})$ to S
        end for
      end for
    end if
  end for
end for
get q quantile of S: S_q
for $i \leftarrow 1$ to n **do**
  for $j \leftarrow 1$ to n **do**
    if $i \neq j$ **then**
      for $P_{ik} \in \mathbf{P}_i$ **do**
        for $P_{jm} \in \mathbf{P}_j$ **do**
          if $f(P_{ik}, P_{jm}) < S_q$ **then**
            Append P_{ik}, P_{jm} to R
            Remove P_{ik}, P_{jm} from $\mathbf{P}_i, \mathbf{P_j}$
          end if
        end for
      end for
    end if
  end for
end for
return Redundancy repository R, updated patch embeddings $\hat{\mathbf{P}}_i$,i=1,2,...,n.

Algorithm 7 Remove redundancy and testing with top k majority voting

Require: Incoming Patches embeddings $\mathcal{P} = (\mathbf{P}_i)$ with $\mathbf{P}_i = \{P_{ij}\}_{j=1,2,...n_i}$. Similarity Score function f, q quantile of Similar Score repository S: S_q, Redundancy repository R, patches embeddings repository (as BOB barcodes) with labels and patients id $(\hat{\mathbf{B}}_i, L_i, G_i)$,i=1,2,...n

Initialize
for $\mathbf{P}_i \in \mathcal{P}$ **do**
  for $P_{ik} \in \mathbf{P}_i$ **do**
    for Patch embedding $R_j \in R$ **do**
      if $f(P_{ik}, R_j) < S_q$ **then**
        Remove P_{ik} from $\mathbf{P}_i$
        Break
      end if
    end for
  end for
  Get the updated embeddings $\tilde{\mathbf{P}}_i$
  Get BOB barcode $\tilde{\mathbf{B}}_i$ from updated embeddings $\tilde{\mathbf{P}}_i$
  Do majority vote of the labels $L_{(1)}, ..., L_{(k)}$ from k closest BOB barcodes from repository $\hat{\mathbf{B}}_{(1)}, ..., \hat{\mathbf{B}}_{(k)}$ which do not share the same patient id with $\tilde{\mathbf{B}}_i$ and predict label $\hat{L}_i$
end for
return Updated patch embeddings $\tilde{\mathbf{P}}_i$,i=1,2,...,n and predicted labels $\hat{L}_i$

Chapter 6

Conclusion and future directions

In this dissertation, I discuss the application of deep learning techniques and learning theory in different nonparametric statistical problems. In this chapter, I will briefly discuss the drawbacks and limitations of the proposed methods in my dissertation and discuss the possible future directions of my research.

In chapter 2, we propose an algorithm leveraging the neural network test error to detect the change points in an offline setting and ensuring the detection accuracy of the number and location of the change points in the data. However, one of the limitations comes from the nature of offline settings. We have to wait until all the data is available and then start to implement the algorithm and make detections retrospectively. In reality, many scenarios require instant detection and are less tolerant of detection delays. The proposed algorithm has the potential to be extended to the online setting because the detection does not need to be started until all the error is calculated and can be a dynamic process. In future studies, we can focus on transforming the algorithm and theoretical results and adapting them in the online setting as well.

The graph model discussed in chapter 3 leverages both spatial and temporal information and consists of two components, one for predicting the existing node points, which is a common problems in my literature. And another is for predicting the new node response

whose previous data and features are not available. The proposed methodology also comes with a theoretical analysis of the convergence rate and experiments on two real datasets. Currently, the model simply add the spatial and temporal information. In future studies, the improvement and modification of the model structure are necessary to keep improving the model performance and let it capable of handling more complicated datasets. For example, we can always incorporate a multilayer structure where each layer is the combination of temporal and spatial messages.

The two step estimator that is proposed in chapter 4 combines the nuclear penalization and the neural network truncation and is robust to various data types in both synthetic and real scenarios. One of the potential drawbacks is that the truncation step loses too much information in the tail. Some contamination mechanisms, like heavy distribution contamination, usually only corrupt a very small proportion of the data. The truncation step, however, just clips all the information, including the useful information. That is part of the reason why the algorithm does not get performance as good as in other data contamination mechanisms. In the future study, I would like to modify the network structure in the truncation step and include some traditional frameworks like Huber loss, specific to the Heavy distribution contamination cases.

Appendices

Appendix A

Supplement for chapter 2

A.1 Supplement of paper: Neural Network-Based Change Point Detection for Large-Scale Time-Evolving Data

A.1.1 β-Hölder smoothness

First, we need some assumptions and also demonstrate the features of the model functions f_j which are composed of several functions: $f_j = g_q \circ g_{q-1} \circ ... \circ g_1 \circ g_0$. We need to define a type of function called β-Hölder smoothness whos partial derivatives exist up to $\lfloor\beta\rfloor$ order and are bounded and the partial derivatives of order $\lfloor\beta\rfloor$ are $\beta - \lfloor\beta\rfloor$ Hölder. Then define the ball of β-Hölder smoothness functions with radius K as:

$$\mathcal{C}_r^{\beta}(D,K) = \{f : D \subset R^r \to R; \sum_{\boldsymbol{\alpha}:|\boldsymbol{\alpha}|<\beta} ||\partial^{\boldsymbol{\alpha}} f||_\infty + \sum_{\boldsymbol{\alpha}:|\boldsymbol{\alpha}|=\lfloor\beta\rfloor} \sup_{x,y\in D;x\neq y} \frac{|\partial^{\boldsymbol{\alpha}} f(x) - \partial^{\boldsymbol{\alpha}} f(y)|}{|x-y|_\infty^{\beta-\lfloor\beta\rfloor}} \leq K\} \quad \text{(A.1)}$$

where $\partial^{\boldsymbol{\alpha}} = \partial^{\alpha_1}...\partial^{\alpha_r}$ with $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_r) \in \mathbb{N}^r$ where ∂^{α_i} denotes the α_i partial derivative on the i^{th} dimension and $|\boldsymbol{\alpha}| := |\boldsymbol{\alpha}|_1$. We assume g_{ij} is β-Hölder smoothness then we define

a function space to be:

$$\mathcal{G}(q, \boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\beta}, K) := \{f = g_q \circ g_{q-1} \circ ... \circ g_1 \circ g_0 : g_i = (g_{ij})_j : [a_i, b_i]^{d_i} \to [a_{i+1}, b_{i+1}]^{d_{i+1}},$$
$$g_{ij} \in \mathcal{C}_{t_i}^{\beta_i}([a_i, b_i]^{t_i}, K) \text{ where } |a_i| \leq K, |b_i| \leq K\} \quad \text{(A.2)}$$

where $\boldsymbol{d} := (d_0, ..., d_{q+1}), \boldsymbol{t} := (t_0, ..., t_q), \boldsymbol{\beta} := (\beta_0, ..., \beta_q)$. This function space is the domain of our model function.

A.1.2 The function space of the neural network

Next, we define the function space of the neural network functions. First, we define a network function space with bounded parameters as:

$$\mathcal{F}(L, \boldsymbol{p}) := \{f : \mathbb{R}^{p_0} \to \mathbb{R}^{p_{L+1}}, x \to f(x) = W_L \sigma_{v_L} W_{L-1} \sigma_{v_{L-1}} ... W_1 \sigma_{v_1} W_0 x :$$
$$\max_{j=0,...,L} ||W_j||_\infty \vee |v_j|_\infty \leq 1\}. \quad \text{(A.3)}$$

where $\boldsymbol{p} = (p_0, ..., p_{L+1}) \in \mathbb{N}^{L+2}$, L is the number of layers of the network, W_j is the $p_{j+1} \times p_j$ matrix, $v_j \in R^{p_j}$ and $\sigma_{v_j}(x) = \max(0, x - v_j)$ is the ReLU function. Then $f : \mathbb{R}^{p_0} \to \mathbb{R}^{p_{L+1}}$ defined above is a neural network with ReLU activation. Here $p_0 = l$ and $p_{L+1} = h$.

Let $|||f|_\infty||_\infty$ denote the sup-norm of the function $x \to |f(x)|_\infty$. Adding a sparsitu condition for the parameters to define the neural network function space as:

$$\mathcal{F}(L, \boldsymbol{p}, s) := \mathcal{F}(L, \boldsymbol{p}, s, F) := \{f \in \mathcal{F}(L, \boldsymbol{p}) : \sum_{j=0}^{L} ||W_j||_0 + |v_j|_0 \leq s, |||f|_\infty||_\infty \leq F\} \quad \text{(A.4)}$$

The following quantities play an important role in the technical developments. First, let

$$\beta_i^* := \prod_{i+1}^{q} (\beta_l \wedge 1) \quad \text{(A.5)}$$

and subsequently define

$$\phi_n := \max_{i=0,\ldots,q} n^{-\frac{2\beta_i^*}{\beta_i^*+t_i}} \tag{A.6}$$

where n is the size of the training set of the neural network.

A.1.3 Useful Lemmas

The following two lemmas in [65, 80] are leveraged in the proof of the main theorems. The model function space and the network function space will be defined as above.

Lemma 5. *Let $f_0 \in \mathcal{G}(q, \boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\beta}, K)$ and define a network function space $\mathcal{F}(L, \boldsymbol{p}, s, F)$.Let $Y_i = f_0(X_i)+\epsilon_i$ as defined by model2.1 where X is independent and$\hat{f} \in arg\min_{f\in\mathcal{F}(L,\boldsymbol{p},s,F)} \sum_{i=1}^n (Y_i - f(X_i))^2$be an empirical risk minimizer and satisfied conditions below:*

(i)$F \geq \max(K, 1)$

(ii)$\sum_{i=0}^q \log_2(4t_i \vee 4\beta_i) \log_2 n \leq L \lesssim n\phi_n$

(iii)$n\phi_n \lesssim \min_{i=1,\ldots,L} p_i$

(iv)$s \simeq n\phi_n \log n$

There exists a constant C' only depending on q,$\boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\beta}$, F such that

$$R(\hat{f} - f_0) = \mathbb{E}_{f_0}[(\hat{f}(X) - f_0(X))^2] \leq C'\phi_n L \log^2 n$$

Lemma 6. *Let $f_0 \in \mathcal{G}(q, \boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\beta}, K)$ and define a network function space $\mathcal{F}(L, \boldsymbol{p}, s, F)$.Let $Y_i = f_0(X_i) + \epsilon_i$ as defined by equation2.1 in section 2.2 where X is dependent and$\hat{f} \in arg\min_{f\in\mathcal{F}(L,\boldsymbol{p},s,F)} \sum_{i=1}^n (Y_i - f(X_i))^2$be an empirical risk minimizer and satisfied conditions below:*

(i)$F \geq \max(K, 1)$

(ii)$\sum_{i=0}^q \log_2(4t_i \vee 4\beta_i) \log_2 n \leq L \lesssim n\phi_n$

(iii)$n\phi_n \lesssim \min_{i=1,\ldots,L} p_i$

(iv)$s \simeq n\phi_n \log n$

There exists a constant C' only depending on q, $\boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\beta}$*, F such that*

$$R(\hat{f} - f_0) = \mathbb{E}_{f_0}[(\hat{f}(X) - f_0(X))^2] \leq C' \phi_n L \log^6 n$$

For the proof of our theorems, we also need 3 common concentration inequalities:

Lemma 7. *Given* $X_1, ..., X_n$ *i.i.d subgaussian, then for* $\forall \delta > 0$*, there exists constant* C_1, C_2 *greater than 0.*

$$P\left(|\bar{X} - \mu| > \delta\right) \leq C_1 e^{-C_2 n}$$

where $\mu = \mathbb{E}[X_i]$ *and* $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$

Lemma 8. *Given* $X_1, ..., X_n$ *independent and from the same distribution, then if* $\mathbb{E}(|X_i|^p) < \infty$*, there exists a constant C>0,*

$$P\left(|\bar{X} - \mu| > \delta\right) \leq \frac{C}{n^{p-1}}$$

where $\mu = \mathbb{E}[X_i]$ *and* $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$

Lemma 9. *Given* $X_1, ..., X_n$ *dependent and from the same distribution, then if* $|cov(X_i, X_j)| < c < 1$ *and* $\mathbb{E}(|X_i|^2) < \infty$*, there exists a constant C>0,*

$$P\left(|\bar{X} - \mu| > \delta\right) \leq \frac{C}{n}$$

where $\mu = \mathbb{E}[X_i]$ *and* $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$

A.1.4 Additional Details Regarding the Proofs of the Main Theorems

Additional details of proof of theorem 1

Similarly, if $\tau > \tau^*$:

$$E(\tau) - E(\tau^*) = \sum_{i=\tau}^{\tau+T_0} ||\hat{f}_\tau^1(X_i) - Y_i||^2 - \sum_{i=\tau^*}^{\tau^*+T_0} ||\hat{f}_{\tau^*}^2(X_i) - Y_i||^2 \geq 0 \quad \text{(A.7)}$$

Where $\hat{f}_\tau^1 \in \arg\min_{f \in \mathcal{F}(L,p,s,F)} \sum_{\tau^* \leq i \leq \tau} (Y_i - f(X_i))^2$ and $\hat{f}_{\tau^*}^2 \in arg\min_{f \in \mathcal{F}(L,p,s,F)} \sum_{i \leq \tau^*} (Y_i - f(X_i))^2$

And then

$$\sum_{i=\tau^*+T_0}^{\tau+T_0} ||\hat{f}_\tau^1(X_i) - Y_i||^2 + \sum_{i=\tau}^{\tau^*+T_0} ||\hat{f}_\tau^1(X_i) - Y_i||^2 - \sum_{i=\tau}^{\tau^*+T_0} ||\hat{f}_{\tau^*}^2(X_i) - Y_i||^2 - \sum_{i=\tau^*}^{\tau} ||\hat{f}_{\tau^*}^2(X_i) - Y_i||^2 \geq 0$$
$$- \sum_{i=\tau^*}^{\tau} ||\hat{f}_{\tau^*}^2(X_i) - Y_i||^2 + \sum_{i=\tau}^{\tau^*+T_0} (||\hat{f}_\tau^1(X_i) - Y_i||^2 - ||\hat{f}_{\tau^*}^2(X_i) - Y_i||^2) + \sum_{i=\tau^*+T_0}^{\tau+T_0} ||\hat{f}_\tau^1(X_i) - Y_i||^2 \geq 0 \quad \text{(A.8)}$$

Here let $I_1 = \sum_{i=\tau^*+T_0}^{\tau+T_0} ||\hat{f}_\tau^1(X_i) - Y_i||^2$, $I_2 = \sum_{i=\tau}^{\tau^*+T_0} (||\hat{f}_\tau^1(X_i) - Y_i||^2 - ||\hat{f}_{\tau^*}^2(X_i) - Y_i||^2)$ and $I_3 = \sum_{i=\tau^*}^{\tau} ||\hat{f}_{\tau^*}^2(X_i) - Y_i||^2$

For I_1:

$$I_1 \leq \sum_{i=\tau^*+T_0}^{\tau+T_0} (||\hat{f}_\tau^1(X_i) - f_1(X_i)||^2 + ||Y_i - f_1(X_i)||^2) \leq (\tau^* - \tau)\mathbb{E}_{f_1}[(\hat{f}_\tau^1(X) - f_1(X)^2]$$
$$+ o(\frac{1}{\tau - \tau^*}) + \sum_{\tau^*+T_0}^{\tau+T_0} ||Y_i - f_1(X_i)||^2 \overset{Theorem5}{\leq} C_1 \phi_{\tau-\tau^*} L \log^2(\tau - \tau^*)(\tau - \tau^*) + h\sigma^2(\tau - \tau^*)$$
$$\leq C_1 \min_{i=1,\ldots,L} p_i \log^2(\tau - \tau^*) + h\sigma^2(\tau - \tau^*) \leq h\sigma^2(\tau - \tau^*) + (\tau - \tau^*)o(1) \quad \text{(A.9)}$$

For I_2:

$$
\begin{aligned}
I_2 &= \sum_{i=\tau}^{\tau^*+T_0} (||\hat{f}_\tau^1(X_i) - Y_i||^2 - ||\hat{f}_{\tau^*}^2(X_i) - Y_i||^2) \\
&\leq \sum_{i=\tau}^{\tau^*+T_0} (||\hat{f}_\tau^1(X_i) - f_1(X_i)||^2 + ||f_1(X_i) - Y_i||^2) - \sum_{i=\tau}^{\tau^*+T_0} ||\hat{f}_{\tau^*}^2(X_i) - Y_i||^2 \\
&\leq T_0 \mathbb{E}_{f_1}[(\hat{f}_\tau^1(X) - f_1(X))^2] + T_0 h\sigma^2 + o(\frac{1}{T_0}) \\
&\quad - \sum_{i=\tau}^{\tau^*+T_0} ||(f_0(X_i) - f_1(X_i)) + (\hat{f}_{\tau^*}^2(X_i) - f_0(X_i)) + (f_1(X_i) - Y_i)||^2 \\
&\leq T_0 \mathbb{E}_{f_1}[(\hat{f}_\tau^1(X) - f_1(X))^2] + T_0 h\sigma^2 \\
&\quad - \sum_{i=\tau}^{\tau^*+T_0} (||f_0(X_i) - f_1(X_i)||^2 - ||\hat{f}_{\tau^*}^2(X_i) - f_0(X_i)||^2 - ||f_1(X_i) - Y_i||^2) \\
&\overset{Theorem5}{\leq} T_0 C_1 \phi_{\tau-\tau^*} L \log^2(\tau - \tau^*) + 2T_0 h\sigma^2 + C_1 \phi_T L \log^2(T) - T_0 M_2 \\
&\leq T_0 \min_{i=1,\ldots,L} p_i \frac{log^2(\tau - \tau^*)}{\tau - \tau^*} + 2T_0 h\sigma^2 - T_0 M_2 + C_1 \phi_T L \log^2(T) \quad \text{(A.10)}
\end{aligned}
$$

For I_3:

$$
\begin{aligned}
I_3 &= \sum_{i=\tau^*}^{\tau} ||\hat{f}_{\tau^*}^2(X_i) - Y_i||^2 = \sum_{i=\tau^*}^{\tau} ||(\hat{f}_{\tau^*}^2(X_i) - f_0(X_i)) + (f_0(X_i) - f_1(X_i)) + (f_1(X_i) - Y_i)||^2 \\
&\geq \sum_{i=\tau^*}^{\tau} (||f_0(X_i) - f_1(X_i)||^2 - ||\hat{f}_{\tau^*}^2(X_i) - f_0(X_i)||^2 - ||f_1(X_i) - Y_i||^2) \\
&\geq (\tau - \tau^*)\mathbb{E}_{f_0}[(f_0(X) - f_1(X))^2] - (\tau - \tau^*)\mathbb{E}_{f_1}[(\hat{f}_{\tau^*}^2(X) - f_1(X))^2] - (\tau - \tau^*)\mathbb{E}[\epsilon^2] + o(1) \\
&\overset{Theorem5}{\geq} (\tau - \tau^*)(M_1 - h\sigma^2 - C_3 \phi_T L \log^2 T) \quad \text{(A.11)}
\end{aligned}
$$

By the previous equation $-I_3 + I_2 + I_1 \geq 0$

$$(\tau - \tau^*)o(1) + h\sigma^2(\tau - \tau^*) + T_0 \min_{i=1,\ldots,L} p_i \frac{\log^2(\tau - \tau^*)}{\tau - \tau^*} + 2T_0 h\sigma^2 + C_1\phi_T L\log^2(T) - T_0 M_2$$

$$\geq (\tau - \tau^*)(M_1 - h\sigma^2 - C_3\phi_T L\log^2 T)$$

$$|\tau - \tau^*| \leq \frac{2T_0 h\sigma^2 + C_1\phi_T L\log^2(T) - T_0 M_2 + T_0 \min_{i=1,\ldots,L} p_i \frac{\log^2(\tau-\tau^*)}{\tau-\tau^*}}{M_1 - 2h\sigma^2 - C_3\phi_T L\log^2 T - o(1)} \quad \text{(A.12)}$$

Additional details of proof of theorem 2

Details of J_2:

$$\begin{aligned} J_2 &= \sum_{k=\tau_i}^{\tau_i+T_0} ||Y_k - \hat{f}_{\tau_i}(X_k)||^2 - \sum_{k=\tau_i+T_0}^{\tau_i+2T_0} ||Y_k - \hat{f}_{\tau_i+T_0}(X_k)||^2 \\ &\geq \sum_{k=\tau_i}^{\tau_i+T_0} (||f_{i+1}(X_k) - f_i(X_k)||^2 - ||Y_k - f_{i+1}(X_k)||^2) - \sum_{k=\tau_i}^{\tau_i+T_0} ||f_i(X_k) - \hat{f}_{\tau_i}(X_k)||^2 \\ &\quad - \sum_{k=\tau_i+T_0}^{\tau_i+2T_0} (||Y_k - f_{i+1}(X_k)||^2 + ||f_{i+1}(X_k) - \hat{f}_{\tau_i+T_0}(X_k)||^2) \\ &\overset{Theorem5}{\geq} o(1) + M1_i T_0 - C_1 T_0 \phi_{\tau_i - \tau_{i-1}} L\log^2(\tau_i - \tau_{i-1}) - C_2 T_0 \phi_{T_0} L\log^2(T_0) - 2T_0 h\sigma^2 \end{aligned} \quad \text{(A.13)}$$

Details of J_3:

$$\begin{aligned} J_3 &= |\sum_{k=\tau_i-T_0}^{\tau_i} ||Y_k - \hat{f}_{\tau_i-T_0}(X_k)||^2 - \sum_{k=\tau_i-2T_0}^{\tau_i-T_0} ||Y_k - \hat{f}_{\tau_i-2T_0}(X_k)||^2| \\ &\leq \sum_{k=\tau_i-T_0}^{\tau_i} (||Y_k - f_i(X_k)||^2 + ||f_i(X_k) - \hat{f}_{\tau_i-T_0}(X_k)||^2) \\ &\quad + \sum_{k=\tau_i-2T_0}^{\tau_i-T_0} (||Y_k - f_i(X_k)||^2 + ||f_i(X_k) - \hat{f}_{\tau_i-2T_0}(X_k)||^2) \\ &\overset{Theorem5}{\leq} o(1) + C_1 T_0 \phi_{\tau_i - \tau_{i-1} - 2T_0} L\log^2(\tau_i - \tau_{i-1} - 2T_0) \\ &\quad + C_2 T_0 \phi_{\tau_i - \tau_{i-1} - T_0} L\log^2(\tau_i - \tau_{i-1} - T_0) + 2T_0 h\sigma^2 \end{aligned} \quad \text{(A.14)}$$

Details of J_4: For J_4:

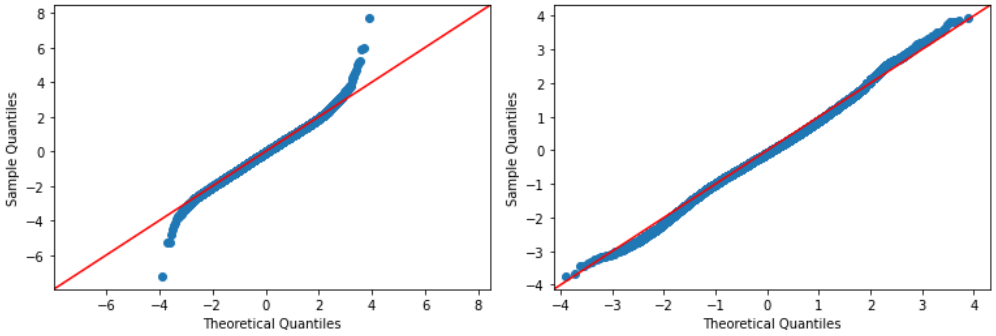
$$
\begin{aligned}
J_4 = |\sum_{k=\tau_i+T_0}^{\tau_i+2T_0} ||Y_k - \hat{f}_{\tau_i+T_0}(X_k)||^2 - \sum_{k=\tau_i+2T_0}^{\tau_i+3T_0} ||Y_k - \hat{f}_{\tau_i+2T_0}(X_k)||^2| \\
&\leq \sum_{k=\tau_i+T_0}^{\tau_i+2T_0} (||Y_k - f_{i+1}(X_k)||^2 + ||f_{i+1}(X_k) - \hat{f}_{\tau_i+T_0}(X_k)||^2) \\
&+ \sum_{k=\tau_i+2T_0}^{\tau_i+3T_0} (||Y_k - f_{i+1}(X_k)||^2 + ||f_{i+1}(X_k) - \hat{f}_{\tau_i+2T_0}(X_k)||^2) \\
&\overset{Theorem5}{\leq} o(1) + 2T_0 h\sigma^2 + C_1 T_0 \phi_{T_0} L \log^2 T_0 + C_2 T_0 \phi_{2T_0} L \log^2(2T_0) \quad \text{(A.15)}
\end{aligned}
$$

Proof of Case II and Case III: For independent and subgaussian function output(*Case II:*), for each changepoint, with probability at least $1 - Ce^{-C_1 T_0}$, the algorithm will detect the changepoint. Then, with probability at least $(1 - Ce^{-C_1 T_0})^N$, the algorithm will detect all N changepoints and exactly these N changepoints and the distance between the estimated change point and the true one will be controlled.

$$
\begin{aligned}
(1 - Ce^{-C_1 T_0})^N \geq 1 - CNe^{-C_1 T_0} &\geq 1 - C_0 \frac{T_{\text{sum}}}{Te^{C_1 T_0}} \\
&\geq 1 - \frac{CT_{\text{sum}}}{C_2 \log(T_{\text{sum}}) T_{\text{sum}}} = 1 - \frac{C_4}{\sqrt{\log(T_{\text{sum}})}} \to 1 \quad \text{(A.16)}
\end{aligned}
$$

For dependent variables*Case III:*, for each changepoint, with probability at least $1 - \frac{C}{T_0}$, the algorithm will detect the changepoint. Then, with probability at least $(1 - \frac{C}{T_0})^N$, the algorithm will detect all N changepoints and exactly these N changepoints and the distance between the estimated change point and the true one will be controlled.

$$
\begin{aligned}
(1 - \frac{C}{T_0})^N \geq 1 - \frac{CN}{T_0} \geq 1 - \frac{CT_{\text{sum}}}{TT_0} \geq 1 - \frac{C_0 T_{\text{sum}}}{T^{1+\rho}} \\
&\geq 1 - \frac{C_0 T_{\text{sum}}}{\frac{\rho+1}{T_{\text{sum}}^{1+\psi\rho}}} \geq 1 - C_0 T_{\text{sum}}^{-\frac{(1-\psi)\rho}{\psi\rho+1}} \underset{T_{\text{sum}} \to \infty}{\to} 1 \quad \text{(A.17)}
\end{aligned}
$$



(a) qqplot for nonsubgaussian data (b) qqplot for subgaussian data

Figure A.1: QQplot

A.1.5 Deviation from normality of the model output

The following figure illustrates the normality of the data outputs: the Q-Q plot on the left corresponds to the non-sub-Gaussian data, while the plot on the right represents the sub-Gaussian case.

Bibliography

[1] Ibrahim Ahmed and Manar Kashmola. *Threats on Machine Learning Technique by Data Poisoning Attack: A Survey*, pages 586–600. Advances in Cyber Security, 01 2021.

[2] Arwa Alanqary, Abdullah Omar Alomar, and Devavrat Shah. Change point detection via multivariate singular spectrum analysis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[3] Areej Alsaafin, Peyman Nejat, Abubakr Shafique, Jibran Khan, Saghir Alfasly, Ghazal Alabtah, and Hamid Tizhoosh. Splice - streamlining digital pathology image processing. *The American Journal of Pathology*, 194, 07 2024.

[4] Craig F Ansley and Robert Kohn. A note on reparameterizing a vector autoregressive moving average model to enforce stationarity. *J. Stat. Comput. Simul.*, 24(2):99–106, June 1986.

[5] Adnan Anwar, Abdun Mahmood, and Mark Pickering. Modeling and performance evaluation of stealthy false data injection attacks on smart grid in the presence of corrupted measurements. *Journal of Computer and System Sciences*, 05 2016.

[6] Orly Ardon, Sylvia L. Asa, Mark C. Lloyd, Giovanni Lujan, Anil Parwani, Juan C. Santa-Rosario, Bryan Van Meter, Jennifer Samboy, Danielle Pirain, Scott Blakely, and Matthew G. Hanna. Understanding the financial aspects of digital pathology: A

dynamic customizable return on investment calculator for informed decision-making. *Journal of Pathology Informatics*, 15:100376, 2024.

[7] Orly Ardon, Eric Klein, Allyne Manzo, Lorraine Corsale, Christine England, Allix Mazzella, Luke Geneslaw, John Philip, Peter Ntiamoah, Jeninne Wright, et al. Digital pathology operations at a tertiary cancer center: Infrastructure requirements and operational cost. *Journal of Pathology Informatics*, 14:100318, 2023.

[8] Orly Ardon, Eric Klein, Allyne Manzo, Lorraine Corsale, Christine England, Allix Mazzella, Luke Geneslaw, John Philip, Peter Ntiamoah, Jeninne Wright, Sahussapont Joseph Sirintrapun, Oscar Lin, Kojo Elenitoba-Johnson, Victor E. Reuter, Meera R. Hameed, and Matthew G. Hanna. Digital pathology operations at a tertiary cancer center: Infrastructure requirements and operational cost. *Journal of Pathology Informatics*, 14:100318, 2023.

[9] Sylvain Arlot, Alain Celisse, and Zaid Harchaoui. A kernel multiple change-point algorithm via model selection. *Journal of Machine Learning Research*, 20(162):1–56, 2019.

[10] Mirko Armillotta and Konstantinos Fokianos. Count network autoregression. *Journal of time series analysis*, 45(4):584–612, 2024.

[11] Kimberly Ashman, Huimin Zhuge, Erin Shanley, Sharon Fox, Shams Halat, Andrew Sholl, Brian Summa, and J. Quincy Brown. Whole slide image data utilization informed by digital diagnosis patterns. *Journal of Pathology Informatics*, 13:100113, 2022.

[12] Jushan Bai and Pierre Perron. Estimating and testing linear models with multiple structural changes. *Econometrica*, pages 47–78, 1998.

[13] Peiliang Bai, Abolfazl Safikhani, and George Michailidis. Multiple change points detection in low rank and sparse high dimensional vector autoregressive models. *IEEE Transactions on Signal Processing*, 68:3074–3089, 2020.

[14] Federal Reseve Bank. National accounts. Technical report, Federal Reserve, 2020. url https://fred.stlouisfed.org/series/A939RX0Q048SBEA .

[15] Monika Bhattacharjee, Moulinath Banerjee, and George Michailidis. Change point estimation in a dynamic stochastic block model. *Journal of Machine Learning Research*, 21(107):1–59, 2020.

[16] Kevin Bleakley and Jean-Philippe Vert. The group fused lasso for multiple change-point detection. *arXiv: Quantitative Methods*, 2011.

[17] Toby Bloom, Rob NicoI, Robert S. Fulton, Michael D. McLellan, John W. Wallis, David E. Larson, Lucinda L. Fulton, Daniel C. Koboldt, John R. Osborne, Chris Markovic, Gary W. Swift, William Courtney, Craig PohI, Scott Abbott, Amy E. Hawkins, Carrie, Haipek, Tammi L. Vickery, Sacha Scott, David J. Dooling, George M. Weinstock, and Jill P. Mesirov. Comprehensive genomic characterization defines human glioblastoma genes and core pathways. In *Cancer Genome Atlas Research Network.*, 2008.

[18] Jerome V. Braun, Ruedi K. Braun, and Hans-Georg Müller. Multiple changepoint fitting via quasilikelihood, with application to dna sequence segmentation. *Biometrika*, 87:301–314, 2000.

[19] Meshack Linda Bulunga. Change-point detection in dynamical systems using auto-associative neural networks. In *Stellenbosch University*, 2012.

[20] Wei-Cheng Chang, Chun-Liang Li, Yiming Yang, and Barnabás Póczos. Kernel change-point detection with auxiliary deep generative models, 2019.

[21] Chang-Hua Chen and Cheng-Few Lee. Threshold vector autoregressive models with exogenous threshold variables. *Journal of Forecasting*, 31(2):135–149, 2012.

[22] Hao Chen and Nancy Zhang. Graph-based change-point detection. *The Annals of Statistics*, 43(1):139 – 176, 2015.

[23] Richard J Chen, Tong Ding, Ming Y Lu, Drew FK Williamson, Guillaume Jaume, Bowen Chen, Andrew Zhang, Daniel Shao, Andrew H Song, Muhammad Shaban, et al. Towards a general-purpose foundation model for computational pathology. *Nature Medicine*, 2024.

[24] Valeriia Cherepanova, Steven Reich, Samuel Dooley, Hossein Souri, Micah Goldblum, and Tom Goldstein. A deep dive into dataset imbalance and bias in face identification, 2022.

[25] Haeran Cho. Change-point detection in panel data via double cusum statistic. *Electronic Journal of Statistics*, 2016.

[26] Haeran Cho and Piotr Fryzlewicz. Multiple-change-point detection for high dimensional time series via sparsified binary segmentation. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 77(2):475–507, July 2014.

[27] Thomas Chong, M. Fernando Palma-Diaz, Craig Fisher, Dorina Gui, Nora L. Ostrzega, Geoffrey Sempa, Anthony E. Sisk, Mark Valasek, Beverly Y. Wang, Jonathan Zuckerman, Chris Khacherian, Scott Binder, and W. Dean Wallace. The california telepathology service: Ucla's experience in deploying a regional digital pathology subspecialty consultation network. *Journal of Pathology Informatics*, 10(1):31, 2019.

[28] Miklós Csörgő and Lajos Horváth. 20 nonparametric methods for changepoint problems. *Handbook of statistics*, 7:403–425, 1988.

[29] Zahra Ebrahimzadeh, Min Zheng, Selcuk Karakas, and Samantha Kleinberg. Pyramid recurrent neural networks for multi-scale change-point detection. In *ICLR 2019 Conference*, 2018.

[30] Majid Ezzati, Hilarie Martin, Suzanne Skjold, Stephen Vander Hoorn, and Christopher Murray. Trends in national and state-level obesity in the usa after correction for self-report bias: Analysis of health surveys. *Journal of the Royal Society of Medicine*, 99:250–257, 05 2006.

[31] Yingzheng Fan, Xingyu Wang, Thomas Funk, Ishrat Rashid, Brianna Herman, Nefeli Bompoti, MD Shaad Mahmud, Maria Chrysochoou, Meijian Yang, Timothy M Vadas, et al. A critical review for real-time continuous soil monitoring: Advantages, challenges, and perspectives. *Environmental Science & Technology*, 56(19):13546–13564, 2022.

[32] Hadi Fanaee-T. Bike Sharing. UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C5W894.

[33] Akylas Fotiadis, Ioannis Vlachos, and Dimitris Kugiumtzis. Detecting nonlinear interactions in complex systems: Application in financial markets. *Entropy*, 25(2):370, 2023.

[34] Piotr Fryslewicz. Wild binary segmentation for multiple change point detection. *The Annals of Statistics*, 42(6):2243–2281, 2014.

[35] Garreau and Arlot. Consistent change-point detection with kernels. *Electronic Journal of Statistics*, 12, 12 2016.

[36] Andreas V Georgiou. The manipulation of official statistics as corruption and ways of understanding it. *Statistical Journal of the IAOS*, 37(1):85–105, 2021.

[37] Jesus Gonzalo and Jean-Yves Pitarakis. Multiple regimes in threshold vector autoregression models. *Econometric Theory*, 21(5):973–1006, 2005.

[38] Muktesh Gupta, Rajesh Wadhvani, and Akhtar Rasool. Real-time change-point detection: A deep neural network-based adaptive approach for detecting changes in multivariate time series data. *Expert Systems with Applications*, 209:118260, 2022.

[39] James D Hamilton. A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica: Journal of the Econometric Society*, pages 357–384, 1989.

[40] Frank Hampel. *Robust Inference*, chapter 1, page 1. John Wiley & Sons, Ltd, 2006.

[41] Frank R. Hampel. *Introduction to Huber (1964) Robust Estimation of a Location Parameter*, pages 479–491. Springer New York, New York, NY, 1992.

[42] Matthew G Hanna, Orly Ardon, Victor E Reuter, Sahussapont Joseph Sirintrapun, Christine England, David S Klimstra, and Meera R Hameed. Integrating digital pathology into clinical practice. *Modern Pathology*, 35(2):152–164, 2022.

[43] Matthew G. Hanna, Victor E. Reuter, Jennifer Samboy, Christine England, Lorraine Corsale, Samson W. Fine, Narasimhan P. Agaram, Evangelos Stamelos, Yukako Yagi, Meera Hameed, David S. Klimstra, and S. Joseph Sirintrapun. Implementation of digital pathology offers clinical and operational increase in efficiency and cost savings. *Archives of Pathology & Laboratory Medicine*, 143(12):1545–1555, 06 2019.

[44] Zaïd Harchaoui and Céline Lévy-Leduc. Multiple change-point estimation with a total variation penalty. *Journal of the American Statistical Association*, 105:1480 – 1493, 2010.

[45] Peter Huber and Elvezio Ronchetti. Robust statistics. 2nd revised ed. *Robust Statistics*, 01 2009.

[46] Peter J. Huber. The 1972 wald lecture robust statistics: A review. *The Annals of Mathematical Statistics*, 43(4):1041–1067, 1972.

[47] Peter J. Huber. John w. tukey's contributions to robust statistics. *The Annals of Statistics*, 30(6):1640–1648, 2002.

[48] P.J. Huber. *Robust Statistics*. Wiley Series in Probability and Statistics - Applied Probability and Statistics Section Series. Wiley, 2004.

[49] Nicholas A. James and David S. Matteson. ecp: An r package for nonparametric multiple change point analysis of multivariate data. *Journal of Statistical Software*, 62(7):1–25, 2015.

[50] S. Kalra, Charles Choi, S. K. Shah, Liron X Pantanowitz, and Hamid R. Tizhoosh. Yottixel - an image search engine for large archives of histopathology whole slide images. *ArXiv*, abs/1911.08748, 2019.

[51] Lucy Kendrick, Katarzyna Musial, and Bogdan Gabrys. Change point detection in social networks - critical review with experiments. *Comput. Sci. Rev.*, 29:1–13, 2018.

[52] R. Killick, P. Fearnhead, and I. A. Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598, oct 2012.

[53] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[54] Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. Statistically significant detection of linguistic change. In *Proceedings of the 24th International World Wide Web Conference*, WWW '15, 2015.

[55] Tze Leung Lai. Sequential changepoint detection in quality control and dynamical systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(4):613–644, 1995.

[56] Stephen E. Lapinsky and Anthony C. Easty. Electromagnetic interference in critical care. *Journal of Critical Care*, 21(3):267–270, 2006.

[57] Junghwan Lee, Yao Xie, and Xiuyuan Cheng. Training neural networks for sequential change-point detection. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023.

[58] Jie Li, Paul Fearnhead, Piotr Fryzlewicz, and Tengyao Wang. Automatic change-point detection in time series via deep learning. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 86(2):273–285, January 2024.

[59] Jun Li, Devin Absher, Hua Tang, Audrey Southwick, Amanda Casto, Sohini Ramachandran, Howard Cann, Gregory Barsh, Marcus Feldman, Luigi Cavalli-Sforza, and Richard Myers. Worldwide human relationships inferred from genome-wide patterns of variation. *Science (New York, N.Y.)*, 319:1100–4, 03 2008.

[60] Siqi Liu, Adam Wright, and Milos Hauskrecht. Change-point detection method for clinical decision support system rule monitoring. *Artificial Intelligence in Medicine*, 91:49–56, 2018.

[61] Ming Chien Lo. Threshold cointegration and nonlinear adjustment to the law of one price. *Economic Modelling*, 25(6):1204–1215, 2008.

[62] Malte Londschien, Peter Bühlmann, and Solt Kovács. Random forests for change point detection, 2023.

[63] Alessandro Longa, Emanuele Rossi, Nesreen K. Ahmed, Michael M. Bronstein, Benjamin Paul Chamberlain, Fabrizio Frasca, and other contributors. Graph neural networks for temporal graphs: State of the art, open challenges, and opportunities. *arXiv preprint arXiv:2302.01018*, 2023.

[64] Helmut Lütkepohl, Timo Teräsvirta, and Jürgen Wolters. Smooth transition vector autoregressive modeling of gdp growth and inflation. *Studies in Nonlinear Dynamics & Econometrics*, 8(1), 2004.

[65] Mingliang Ma and Abolfazl Safikhani. Theoretical analysis of deep neural networks for temporally dependent observations. *Advances in Neural Information Processing Systems*, 35:37324–37334, 2022.

[66] Ričards Marcinkevičs and Julia E. Vogt. Interpretable models for granger causality using self-explaining neural networks, 2021.

[67] R.A. Maronna, R.D. Martin, V.J. Yohai, and M. Salibián-Barrera. *Robust Statistics: Theory and Methods (with R)*. Wiley Series in Probability and Statistics. Wiley, 2019.

[68] Sohir Maskey, Ron Levie, Yunseok Lee, and Gitta Kutyniok. Generalization analysis of message passing neural networks on large random graphs. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 4805–4817. Curran Associates, Inc., 2022.

[69] George Michailidis. Miami housing dataset, 2024. Confidential data, not publicly available.

[70] Yue S. Niu and Heping Zhang. The screening and ranking algorithm to detect DNA copy number variations. *The Annals of Applied Statistics*, 6(3), sep 2012.

[71] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org. https://www.openstreetmap.org, 2024. OpenStreetMap is open data, licensed under the Open Data Commons Open Database License (ODbL).

[72] Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. Graph neural ordinary differential equations. *arXiv preprint arXiv:1911.07532*, 2019.

[73] Adarsh Prasad, Arun Sai Suggala, Sivaraman Balakrishnan, and Pradeep Ravikumar. Robust estimation via robust gradient estimation, 2018.

[74] S.T. Rachev. *Handbook of Heavy Tailed Distributions in Finance: Handbooks in Finance, Book 1.* Handbooks in Finance. North Holland, 2003.

[75] Noah A. Rosenberg, Jonathan K. Pritchard, James L. Weber, Howard M. Cann, Kenneth K. Kidd, Lev A. Zhivotovsky, and Marcus W. Feldman. Genetic structure of human populations. *Science*, 298(5602):2381–2385, 2002.

[76] Robert Rosenman, Vidhura Tennekoon, and Laura Hill. Measuring bias in self-reported data. *Int. J. of Behavioural and Healthcare Research*, 2:320 – 332, 10 2011.

[77] Emanuele Rossi, Benjamin Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

[78] Peter Rousseeuw. Tutorial to robust statistics. *Journal of Chemometrics*, 5:1 – 20, 01 1991.

[79] Abolfazl Safikhani, Yue Bai, and George Michailidis. Fast and scalable algorithm for detection of structural breaks in big VAR models. *J. Comput. Graph. Stat.*, 31(1):176–189, 2022.

[80] Johannes Schmidt-Hieber. Nonparametric regression using deep neural networks with ReLU activation function. *The Annals of Statistics*, 48(4), aug 2020.

[81] M.A. Seligson and D.E. Morales. *Improving the qualit y of survey data using capi systems in developing countries*, pages 207–219. The Oxford University Press, 01 2015.

[82] Abubakr Shafique, Saghir Alfasly, Areej Alsaafin, Peyman Nejat, Jibran A. Khan, and H. R. Tizhoosh. Selection of distinct morphologies to divide & conquer gigapixel pathology images, 2023.

[83] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified defenses for data poisoning attacks, 2017.

[84] George C Tiao and Ruey S Chen. Vector smooth transition autoregressive models for nonlinear multivariate time series. *Working Paper*, 2000.

[85] Michalis K. Titsias, Jakub Sygnowski, and Yutian Chen. Sequential changepoint detection in neural networks with checkpoints, 2020.

[86] H. R. Tizhoosh, Shujin Zhu, Hanson Lo, Varun Chaudhari, and Tahmid Mehdi. Minmax radon barcodes for medical image retrieval, 2016.

[87] Hamid R Tizhoosh and Liron Pantanowitz. On image search in histopathology. *Journal of Pathology Informatics*, page 100375, 2024.

[88] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems, 2020.

[89] Howell Tong. *Non-linear Time Series: A Dynamical System Approach*. Oxford University Press, 1990.

[90] Charles Truong, Laurent Oudre, and Nicolas Vayatis. Selective review of offline change point detection methods. *Signal Processing*, 167:107299, February 2020.

[91] Ruey S Tsay. Testing and modeling multivariate threshold models. *Journal of the American Statistical Association*, 93(443):1188–1202, 1998.

[92] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[93] Shuoyang Wang and Guanqun Cao. Robust deep neural network estimation for multidimensional functional data, 2022.

[94] Xiaolei Wang. Aqi and weather data in china. https://quotsoft.net/air/, 2021.

[95] Yunli Wang and Cyril Goutte. Real-time change point detection using on-line topic models. In *International Conference on Computational Linguistics*, 2018.

[96] Mark Woodard, Sahra Sedigh, and A. R. Hurson. *A Survey of Research on Data Corruption in Cyber-Physical Critical Infrastructure Systems*, chapter 98, pages 59–87. Advances in Computers, January 2015.

[97] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S. Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2020.

[98] Shiyang Yan, Yue Xiong, and Dahua Lin. Temporal shift graph attention networks for time series forecasting. In *Advances in Neural Information Processing Systems*, volume 35, pages 22937–22949, 2022.

[99] Hang Yin, Abolfazl Safikhani, and George Michailidis. A general modeling framework for network autoregressive processes. *Technometrics*, 65(4):579–589, 2023.

[100] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3634–3640, 2018.

[101] Xuezhou Zhang, Xiaojin Zhu, and Laurent Lessard. Online data poisoning attack, 2019.

[102] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianliang Qi. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1234–1241, 2020.

[103] Xiaoqian Zhu, Yongjia Xie, Jianping Li, and Dengsheng Wu. Change point detection for subprime crisis in american banking: From the perspective of risk dependence. *International Review of Economics & Finance*, 38:18–28, 2015.

[104] Xuening Zhu, Rui Pan, Guodong Li, Yuewen Liu, and Hansheng Wang. Network vector autoregression. *The Annals of Statistics*, 45(3):1096–1123, 2017.

[105] Ángel Carmona-Poyato, Nicolás Luis Fernández-Garcia, Francisco José Madrid-Cuevas, and Antonio Manuel Durán-Rosal. A new approach for optimal offline time-series segmentation with error bound guarantee. *Pattern Recognition*, 115:107917, 2021.