SPATIO-TEMPORAL COPULA-BASED GRAPH NEURAL NETWORKS FOR TRAFFIC FORECASTING

Mr. Pitikorn Khlaisamniang

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Statistics

Department of Statistics

Faculty of Commerce and Accountancy

Chulalongkorn University

Academic Year 2022

โครงข่ายประสาทเทียมแบบกราฟเชิงปริภูมิกาลโดยใช้คอปปูลา สำหรับการพยากรณ์การจราจร

นายปิติกร คล้ายสำเนียง

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาสถิติ ภาควิชาสถิติ
คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2565

Thesis Title	SPATIO-TEMPORAL COPULA-BASED GRAPH NEURAL NETWORKS FOR TRAFFIC FORECASTING
By	Mr. Pitikorn Khlaisamniang
Field of Study	Statistics
Thesis Advisor	Assistant Professor SURONAPEE PHOOMVUTHISARN, Ph.D.

Accepted by the Faculty of Commerce and Accountancy, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

. . . . . Dean of the Faculty of Commerce and Accountancy

(Professor Wilert Puriwat, Ph.D.)

THESIS COMMITTEE

. . . . . Chairman

(Assistant Professor AKARIN PHAIBULPANICH, Ph.D.)

. . . . . Thesis Advisor

(Assistant Professor SURONAPEE PHOOMVUTHISARN, Ph.D.)

. . . . . Examiner

(Assistant Professor NAT KULVANICH, Ph.D.)

. . . . . External Examiner

(CHALEE THAMMARAT, Ph.D.)

ปิติกร คล้ายสำเนียง: โครงข่ายประสาทเทียมแบบกราฟเชิงปริภูมิกาลโดยใช้คอปปูลาสำหรับการพยากรณ์การจราจร. (SPATIO-TEMPORAL COPULA-BASED GRAPH NEURAL NETWORKS FOR TRAFFIC FORECASTING) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ. ดร. สุรณพีร์ ภูมิวุฒิสาร, 0 หน้า.

เมืองสมัยใหม่พึ่งพาการคมนาคมที่ซับซ้อนเป็นอย่างมากดังนั้นการคาดคะเนสภาพการจราจรที่แม่นยำมีความสำคัญอย่างยิ่งสำหรับหน่วยงานด้านการจัดการการจราจร วิธีการดั้งเดิมซึ่งประกอบไปด้วยวิธีการเชิงสถิติและวิธีการการเรียนรู้ของเครื่องแบบดั้งเดิมไม่สามารถที่จะตรวจจับความสัมพันธ์ที่ซับซ้อนได้ ในขณะที่การเรียนรู้เชิงลึกยังมีข้อด้อยในเรื่องของการทบกันของความผิดพลาด ความยากลำบากในการจัดการลำดับที่ยาว และการไม่คำนึงถึงความสัมพันธ์เชิงพื้นที่ โครงข่ายประสาทเทียมแบบกราฟได้แสดงให้เห็นถึงความสามารถในการแยกลักษณะเชิงพื้นที่ออกจากโครงสร้างกราฟที่ไม่ใช่แบบยุคลิดแต่โครงข่ายประสาทเทียมแบบกราฟส่วนมากนิยามแมทริกซ์ประชิดจากระยะห่างระหว่างจุดของกราฟซึ่งไม่สามารถตรวจจับความสัมพันธ์ในเชิงสถิติได้ การเลือกวัดความสัมพันธ์ระหว่างจุดของกราฟจึงมีความสำคัญในการทำโมเดลเชิงกราฟ วิทยานิพนธ์เล่มนี้นำเสนอ โครงข่ายประสาทเทียมแบบกราฟเชิงพื้นที่และเวลาแบบหลายมุมมองซึ่งสามารถตรวจจับข้อมูลในเชิงความสัมพันธ์ที่เป็นจริงและความสัมพันธ์เชิงสถิติ ซึ่งแตกต่างจากการวัดความสัมพันธ์แบบดั้งเดิม คอปปูลาสามารถแยกความสัมพันธ์ทางสถิติที่ซ่อนอยู่และสร้างฟังก์ชันการแจกแจงหลายตัวแปรเพื่อให้ได้ความสัมพันธ์ระหว่างจุดของการจราจร มีการนำวิธีการสองขั้นตอนมาใช้ประกอบด้วยการทดสอบคอปปูลาของสองตัวแปรเพื่อหาคอปปูลาที่เหมาะสมในการนิยามความสัมพันธ์ระหว่างคู่จุดใด ๆ ในกราฟของการจราจร และนำพารามิเตอร์ของคอปปูลานั้น ๆ มาสร้างเป็นแมทริกซ์ประชิด ขั้นตอนต่อมาจะเป็นการใช้โมเดลโครงข่ายประสาทเทียมเชิงกราฟแบบคอนโวลูชันในการสกัดข้อมูลเชิงพื้นที่และตรวจจับข้อมูลเชิงเวลาด้วยไดเลตเต็ดคอซอลคอนโวลูชัน โมเดล ST-CopulaGNN ที่ทางผู้จัดทำได้นำเสนอให้ประสิทธิภาพที่ดีกว่าโมเดลก่อนหน้าอย่าง DCRNN และ Graph WaveNet ชี้ให้เห็นถึงผลลัพธ์ของการนำคอปปูลามาใช้ในการวิเคราะห์การจราจรโดยทดลองบนชุดข้อมูล METR-LA และ PEMS-BAY

ภาควิชา	สถิติ	ลายมือชื่อนิสิต	.....
สาขาวิชา	สถิติ	ลายมือชื่อ อ.ที่ปรึกษาหลัก	.....
ปีการศึกษา	2565		

6480449226: MAJOR STATISTICS
KEYWORDS: COPULA / CORRELATION / GRAPH NEURAL NETWORKS / TRAFFIC FORECASTING

PITIKORN KHLAISAMNIANG : SPATIO-TEMPORAL COPULA-BASED GRAPH NEURAL NETWORKS FOR TRAFFIC FORECASTING. ADVISOR : Asst. Prof. SURONAPEE PHOOMVUTHISARN, Ph.D., 0 pp.

Modern cities heavily rely on complex transportation, making accurate traffic speed prediction crucial for traffic management authorities. Classical methods, including statistical techniques and traditional machine learning techniques, fail to capture complex relationships, while deep learning approaches may have weaknesses such as error accumulation, difficulty in handling long sequences, and overlooking spatial correlations. Graph neural networks (GNNs) have shown promise in extracting spatial features from non-Euclidean graph structures, but they usually initialize the adjacency matrix based on distance and may fail to detect hidden statistical correlations. The choice of correlation measure can have a significant impact on the resulting adjacency matrix and the effectiveness of graph-based models. This thesis proposes a novel approach for accurately forecasting traffic patterns by utilizing a multi-view spatio-temporal graph neural network that captures data from both realistic and statistical domains. Unlike traditional correlation measures such as Pearson correlation, copula models are utilized to extract hidden statistical correlations and construct multivariate distribution functions to obtain the correlation relationship among traffic nodes. A two-step approach is adopted, which involves selecting and testing different types of bivariate copulas to identify the ones that best fit the traffic data, and utilizing these copulas to create multi-weight adjacency matrices. The second step involves utilizing a graph convolutional network to extract spatial information and capturing temporal trends using dilated causal convolu-

tions. The proposed ST-CopulaGNN model outperforms previous approaches such as DCRNN and Graph WaveNet, indicating the effectiveness of incorporating copulas in traffic forecasting. Experiments on the METR-LA and PEMS-BAY datasets show that the proposed model outperforms previous approaches with a slight improvement.

Department:	Statistics	Student's Signature	.....
Field of Study:	Statistics	Advisor's Signature	.....
Academic Year:	2022		

Acknowledgements

The work presented in this paper received support from Artificial Intelligence Association of Thailand (AIAT) and NSTDA Supercomputer Center (ThaiSC) for Huawei Cloud credits and the Thai supercomputer Lanta, which is equipped with 704 NVIDIA A100 GPUs.

I would like to thank Assistant Professor Suronapee Phoomvuthisarn, Ph.D., my advisor who always recommends the best method to get through all the obstacles.

CONTENTS

Page

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

Chapter I

BACKGROUND

1.1 Background and Problem Review

Urbanization and population growth have led to increasingly complex transportation networks. These networks have become a crucial component of modern cities infrastructure **?**, with millions of people relying on them for daily commuting and travel. To improve the quality of transportation systems, intelligent transportation systems (ITS) have emerged as an effective approach. ITS incorporates modern wireless, electronic, and automated technologies to integrate users, infrastructure, and vehicles into a seamless and efficient system **?**. Many countries have made significant commitments to developing ITS, which leverage vast amounts of urban traffic data collected from various sources, including road sensors, taxi and private car trajectories, and public transportation transaction records **?**.

Accurate prediction of traffic speed is essential for traffic management authorities to direct vehicles more effectively, improving the smooth operation of the highway network. In the domain of traffic forecasting, numerous studies have been conducted, which can be generally classified into two methodologies: classical methods and deep learning methods. Classical methods include statistical techniques such as Historical Averages (HA), Auto-Regressive Integrated Moving Average (ARIMA), and Vector Auto-Regressive (VAR), and traditional machine learning techniques such as Support Vector Regression (SVR), Random Forest Regression (RFR), and K-Nearest Neighbor (KNN). These models are limited in their ability to capture complex, nonlinear patterns in traffic data. On the other hand, deep learning approaches such as Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs) and their variants have shown promise in capturing complex nonlinear relationships in traffic forecasting. However, RNN-based methodologies can

exhibit limitations, including the accumulation of errors, slow training, and challenges in effectively processing long sequences, while CNNs interpret traffic data as Euclidean data, which limits the amount of spatial correlation they can capture. Several studies have utilized CNNs to capture the spatial-correlation of traffic data, but these algorithms may not be able to fully capture all the spatial correlations present in the data due to the non-Euclidean nature of traffic data and its own topology. In recent years, graph neural networks (GNNs) have been widely employed in traffic speed forecasting due to their efficiency in extracting spatial features represented by non-Euclidean graph structures. For example, a network of roads naturally forms a graph, with intersections serving as nodes and connections as edges. The GNN-based methodology has been expanded to transportation field, making use of different graph formulations and models like the diffusion convolutional recurrent neural network (DCRNN) **?** and Graph WaveNet **?** model.

However, these models usually initialize the adjacency matrix based on distance, which fails to detect hidden statistical correlation. As a result, the choice of correlation measure can have a significant impact on the resulting adjacency matrix, and hence the effectiveness of the graph-based models. In Figure **??**, we show the adjacency matrix from different correlation measures, while Figure **??** represents the linkage between pairs of nodes that are not connected but have a correlation in the statistical domain. These figures demonstrate that a number of correlation measures have been proposed to compute the adjacency matrix, such as Pearson correlation coefficient, Spearman's rank correlation, and Kendall's tau correlation. These methods construct multivariate distribution functions to obtain the correlation relationship among traffic nodes.

The challenge in modeling the dependence between links in a stochastic transportation network is that it requires specifying a complex dependence structure that accurately captures the statistical relationships between the variables, which is crucial for effective learning, inference, and representation of the joint distribution of several links' travel time. To address this challenge, copula models are utilized in

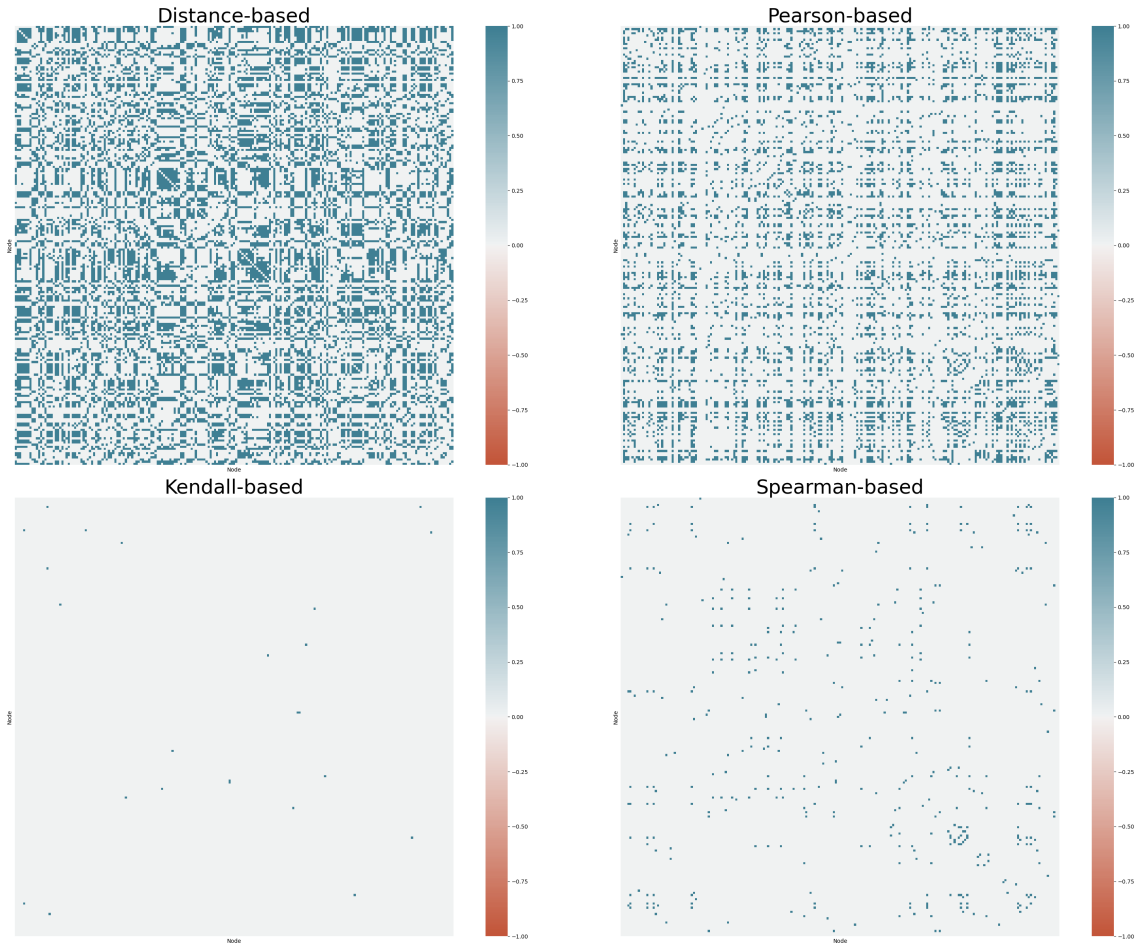


Figure 1.1: Different adjacency matrices from METR-LA (with absolute correlation coefficient > 0.75).

this research. Figure **??** shows the Pearson correlation between sample nodes and scatter plots. Pearson correlation is a measure of the linear association between two variables. It is used to determine how closely related two variables are and the direction of the relationship, but it cannot capture the dependence structure between variables. Unlike traditional correlation measures, copulas provide a modern approach with a rigid structure that can describe not only the relationship between random variables but also any other properties relevant to the entire structure **?**. This becomes feasible since any multivariate joint distribution can be expressed using individual marginal distribution functions and a copula, which characterizes the interdependence between variables, as indicated by Sklar's theorem. By using copula models, hidden statistical correlations can be extracted, and multivariate dis-

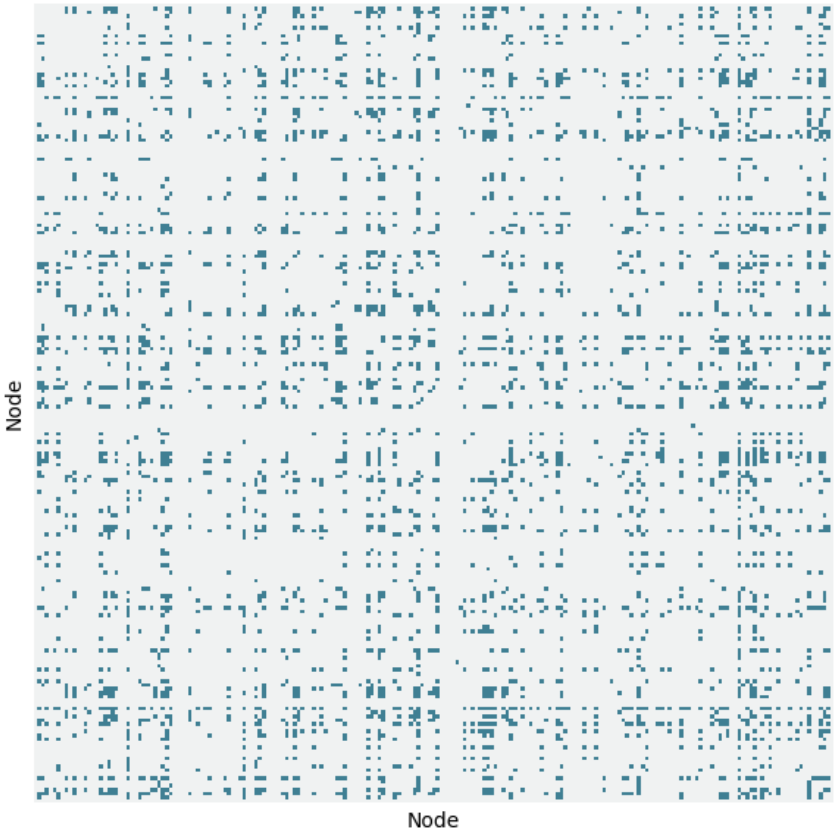


Figure 1.2: Adjacency matrix displays pairs of nodes that exhibit strong correlation despite not being directly connected from METR-LA.

tribution functions can be constructed to obtain the correlation relationship among traffic nodes, leading to more accurate modeling of the stochastic transportation network.

In this thesis, we propose a novel method that employs a multi-view spatio-temporal graph neural network model to capture data from both realistic and statistical domains. In this study, we adopt a two-step approach to address the challenge of capturing real-world scenarios using copulas and designing a copula-based adjacency matrix architecture. The first step involves selecting and testing different types of bivariate copulas (i.e., Gaussian, Clayton, Gumbel, Frank) to identify the ones that best fit the traffic data. We then use these copulas to create multi-weight adjacency matrices that capture the statistical correlations between traffic nodes. The second step involves utilizing a graph convolutional network to extract spatial

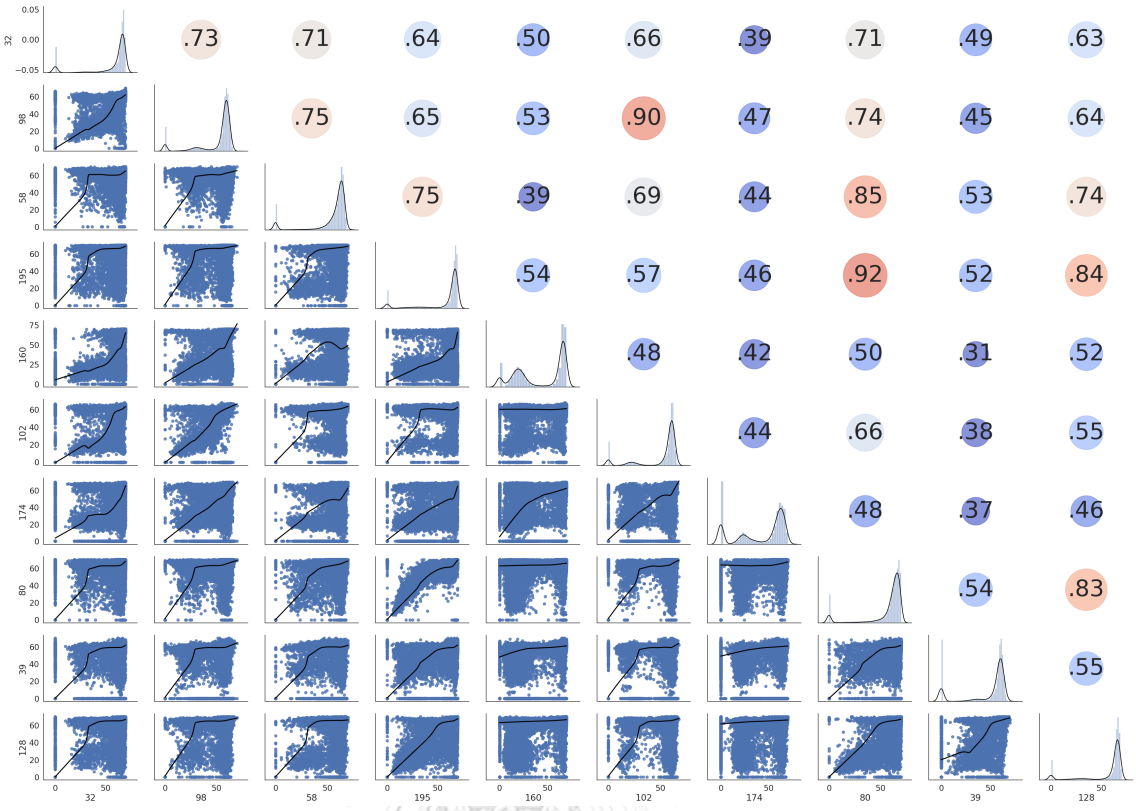


Figure 1.3: Plots showing pairwise comparisons of randomly selected nodes from METR-LA. Scatter plots [lower-left], Histograms [diagonal], Pearson Correlation [upper-right]

information from the preprocessed data and adjacency matrices, and capturing temporal trends using dilated causal convolutions as temporal convolution layer. This approach enables us to train a multi-view spatio-temporal graph neural network model that can accurately forecast traffic patterns.

1.2 Statement of Problem

Traffic forecasting is a time series prediction challenge. The initial adjacency matrix created to serves as the prior knowledge. We can define it as a graph G

$$G = (V, E, A)$$

where:

V is the set of nodes, where each node corresponds to an observation point (such

as a sensor or road segment).

E is the set of edges. In graph G, the edges represent the connections between the nodes, while the edge weights denote the distances between the respective nodes.

$A \in \mathbb{R}^{n \times n}$ is adjacency matrix of graph G

The objective of traffic forecasting is to predict the upcoming traffic sequence. $X_G^{(t+1)}, X_G^{(t+2)}, \ldots, X_G^{(t+T')}$ using a historical data $X_G^{(t-T+1)}, X_G^{(t-T+2)}, \ldots, X_G^{(t)}$

where:

$X_G^{(t)} \in \mathbb{R}^{N \times C}$ is the the observation of graph G at time step t

C is the number of features

T is the length of window to capture historical data and

T' is the prediction length

We can defined the problem for traffic forecast as:

$$\left(X_G^{(t-T+1)}, X_G^{(t-T+2)}, \ldots, X_G^{(t)}\right) \xrightarrow{f} \left(X_G^{(t+1)}, X_G^{(t+2)}, \ldots, X_G^{(t+T')}\right)$$

where:

f is the learning function

From the above equation, the adjacency matrix can be used to define the structure of the road network by determining which roads are connected to each other. However, using distance-based approaches alone to create the initial adjacency matrix can overlook hidden statistical correlations between traffic nodes on the road

network. To address this gap in the literature, our proposed solution utilizes Bivariate Copula to create multi-weight adjacency matrices that can capture not only the correlation between random variables, but also any other significant attributes pertaining to the entire structure of the road network.

1.3 Objectives

This research aims to enhance spatio-temporal model in term of adjacency relation.

Chapter II

RELATED WORK

2.1 Traffic forecast

Numerous studies have been conducted in the field of traffic forecasting, which can be broadly categorized into two methodologies: classical methods and deep learning methods.

Classical methodologies for traffic forecasting include classical statistics and traditional machine learning techniques. Examples of early statistical techniques used for traffic forecasting include Historical Averages (HA) **?**, Auto-Regressive Integrated Moving Average (ARIMA) **?**, and Vector Auto-Regressive (VAR) **?**. These models are limited in their ability to accurately forecast traffic due to their reliance on static assumptions based on linear time series approaches. As a result, they may not be able to capture the complex, nonlinear patterns that are often present in traffic data. In contrast, machine learning-based approaches have shown better performance as they can capture the nonlinear complexity of traffic data.

To capture complex nonlinear relationships, traditional machine learning techniques such as Support Vector Regression (SVR) **?**, Random Forest Regression (RFR) **?**, and K-Nearest Neighbor (KNN) **?** have been applied to traffic forecasting. Although effective, designing manual features for traditional machine learning techniques requires expertise and domain knowledge, and may not capture all relevant information in the data, leading to suboptimal performance.

Deep learning approaches have shown promise in capturing complex nonlinear relationships in traffic forecasting. For example, Recurrent Neural Networks (RNNs), including LSTM and GRU models, have been successfully used to capture the temporal features of traffic data **????**. However, RNN-based approaches may

suffer from weaknesses such as error accumulation, slow training, and difficulty in handling long sequences. Convolutional Neural Networks (CNNs) have gained significant popularity as a favored option for processing data in parallel with low memory overhead. In the domain of time series forecasting, CNN-based techniques such as WaveNet **?** and Temporal Convolutional Neural Network (TCN) **?** have been widely used. However, the above-mentioned approaches overlook the spatial correlation of traffic data as a comprehensive reflection of the spatial-temporal dependence.

Several studies have utilized CNNs to capture the spatial-correlation of traffic data. Zhang et al. presented ST-ResNet **?**, which used a CNN structure to forecast urban pedestrian traffic. Similarly, Yao et al. developed a spatial-temporal dynamic network using CNN and LSTM to forecast New York taxi and cycling data **?**. However, CNNs interpret traffic data as Euclidean data, which limits the amount of spatial correlation they can capture. Since traffic data fundamentally exhibits a non-Euclidean nature and the road network has its own topology, CNN-based algorithms may not be able to capture all the spatial correlations present in the data.

To address this limitation, Graph Neural Networks (GNNs) have been increasingly used to model non-Euclidean data, proven effective in capturing spatial correlations, particularly in spatial-temporal data. In such modelling techniques, spatio-temporal correlation is usually captured through the use of RNNs, CNNs, and attention mechanisms **?**. For example, DCRNN **?** uses GRU along with diffusion GCN to represent the spatial correlation of traffic by considering it as a diffusion process on directed graphs. On the other hand, urban traffic is a system that undergoes dynamic changes, representing such dynamics using a fixed graph structure with a static adjacency matrix is not feasible. To address this limitation and capture dynamic spatial correlation, Wu et al. proposed Graph WaveNet **?**, which uses GCN to create an adaptive adjacency matrix.

The adjacency matrix in Graph WaveNet and DCRNN is established using a

distance-based approach, but statistical methods can also capture correlations between disconnected nodes, as shown in Figure **??**, to extract semantic information. In our proposed model, we differ from Graph WaveNet by utilizing multi-weight adjacency matrices obtained from Bivariate Copula which can capture not only the correlation between random variables but also any other significant attributes pertaining to the entire structure.

2.2 Adjacency Matrix

Adjacency matrix $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ records the linkage between each node pair. If node v_i and node v_j belong to V, and they are connected ($e_{ij} \in E$), then $a_{ij} = 1$, otherwise $a_{ij} = 0$

Distance-based Adjacency Matrix

Previous works **??** defined adjacency matrix from distance-based using thresholded Gaussian kernel as follow:

$$a_{ij} = \begin{cases} \exp\left(-\frac{\text{dist}(v_i, v_j)^2}{\sigma^2}\right) & ; \text{if dist}(v_i, v_j) \leq \kappa \\ 0 & ; \text{otherwise} \end{cases} \tag{2.1}$$

where:

dist(v_i, v_j) represents the distance within the road network from detector v_i to detector v_j.

σ is the standard deviation of distances.

κ is the threshold.

Self-adaptive Adjacency Matrix

Graph WaveNet propose a self-adaptive adjacency matrix $\tilde{A}_{adp}$which does not necessitate any preexisting knowledge and is learned end-to-end through stochastic gradient descent. The model discovers hidden spatial dependencies by itself, which is accomplished by randomly initializing two node embedding dictionaries with adjustable parameters.

$$\tilde{A}_{adp} = \text{SoftMax}(\text{ReLU}(L_1 L_2^T)) \tag{2.2}$$

where:

L_1 is the source node embedding

L_2 is the target node embedding

Pearson correlation-based Adjacency Matrix

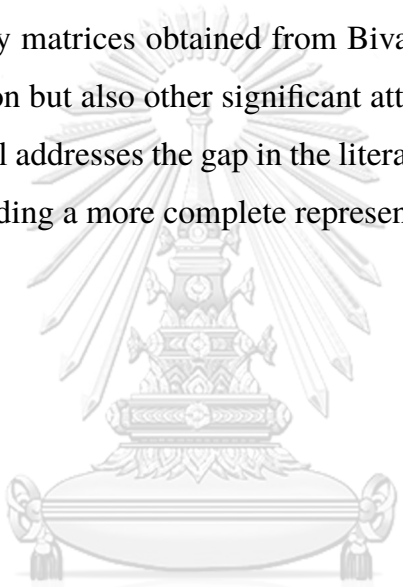
The Pearson correlation coefficient can be utilized as an adjacency matrix since it measures node correlation. TGANet **?** improve Graph Wavenet by applying the Pearson correlation coefficient between each node pair by utilizing their historical data to show hidden correlation between each pair of nodes. Given nodes v_i and v_j for instance, the Pearson correlation coefficient between the historical data E_i and E_j from v_i and v_j can be obtained using Eq. **??**, where $cov(E_i, E_j)$ is the covariance between E_i and E_j, σ_{E_i} and σ_{E_j} are the standard deviations of E_i and E_j, μ_{E_i} and μ_{E_j} are the mean values of E_i and E_j, respectively.

The following is a description of the adjacency matrix with correalation-based:

$$a_{i,j} = \rho_{E_i,E_j} = \frac{\text{cov}(E_i, E_j)}{\sigma_{E_i} \cdot \sigma_{E_j}} = \frac{\text{E}\left[(E_i - \mu_{E_i})(E_j - \mu_{E_j})\right]}{\sigma_{E_i} \cdot \sigma_{E_j}} \tag{2.3}$$

Gaps in literature

The literature on graph-based models such as Graph WaveNet has focused on using distance-based approaches to establish adjacency matrices. However, these methods fail to capture hidden statistical correlations between disconnected nodes, which can provide valuable semantic information. As shown in Figure **??**, statistical methods can be used to extract this information. While Pearson Correlation is a commonly used measure of correlation, it has limitations, such as only being applicable to linear regression models. In contrast, our proposed model utilizes multi-weight adjacency matrices obtained from Bivariate Copula, which can capture not only correlation but also other significant attributes of the entire structure. By doing so, our model addresses the gap in the literature by improving upon existing methods and providing a more complete representation of the data.



Chapter III

RESEARCH METHODOLOGY

3.1 Hypotheses

Copula functions have been extensively employed across various fields, including hydrology, asset pricing, and credit risk management in finance **?**. Which, similar to numerous transportation models that emphasize reliability, this approach also includes the ranking of random variables, taking into account complex correlations. Simply speaking, Pearson correlation is a measure of the linear association between two variables. It is used to determine how closely related two variables are and the direction of the relationship. Pearson correlation ranges from -1 to +1, where a value of -1 indicates a perfect negative linear relationship, +1 indicates a perfect positive linear relationship, and 0 indicates no linear relationship between the two variables.

On the other hand, Copula is a statistical tool that measures the dependence structure between two variables. It is used to analyze the relationship between two variables beyond the linear relationship captured by the Pearson correlation coefficient. Copula allows for a more flexible modeling of the relationship between variables, as it does not require the variables to follow a particular distribution.

While Pearson correlation measures the linear relationship between variables, Copula captures the dependence structure between variables, which can be nonlinear or complex. Copula can be especially useful in analyzing complex relationships between variables, such as those that involve multiple variables or variables that follow non-normal distributions.

Thus, the way to improve better understand in correlation we purpose Copulas which provide a broader perspective than correlation and are supported by a

rigorous theoretical foundation. From **?** leads us to the conclusion that copulas offer significant advantages over ordinary correlation, as they provide a superior and more adaptable framework. Additionally, copulas exhibit greater flexibility due to their lack of restrictions, leading to enhanced computational capabilities. Overall, copulas represent a more comprehensive form of modeling compared to correlation.

3.2 Concept of Copula

In probability theory, a Copula is a function that describes the dependence structure between multiple random variables. More specifically, a Copula is a multivariate distribution function that links the marginal distributions of each variable to the joint distribution of all the variables.

The Copula function has the important property of being able to separate the dependence structure between variables from their individual marginal distributions. This means that we can model the joint distribution of variables using a Copula that takes into account the dependence structure between them, while allowing each variable to have its own marginal distribution.

In other words, a Copula is a way to model the joint distribution of variables that captures their dependence structure in a way that is independent of their marginal distributions. Copulas are useful for a variety of applications, including risk management, finance, insurance, and weather forecasting, to name a few. Sklar **?** introduced the concept of copula as a means to model the dependence among multiple random variables. The concept of copula is described in **?** .

Definition 1 (A d-dimensional copula)**.** $C : [0,1]^d \to [0,1]$ is a cumulative distribution function (CDF) with uniform mar-ginals.

Given $C(u) = C(u_1, u_2, ..., u_d)$ for a generic copula and immediately have the following properties.

1. $C(u_1, u_2, ..., u_d)$ is non-decreasing in each component, u_i.

2. The i^{th} marginal distribution is obtained by setting $u_j = 1$ for $j \neq i$ and since it it is uniformly distributed

$$C(1, ..., 1, u_i, 1, ..., 1) = u_i$$

3. For $a_i \leq b_i, P_i(U_1 \in [a_1, b_1], ..., U_d \in [a_d, b_d])$ must be non-negative. This implies the rectangle inequality

$$\sum_{i_1=1}^{2} \cdots \sum_{i_d=1}^{2} (-1)^{i_1+i_2+...+i_d} \cdot C(u_{1,i_1}, ..., u_{d,i_d}) \geq 0$$

where $u_{j,1} = a_j$ and $u_{j,2} = b_j$

Likewise, every function that meets properties 1 through 3 is a copula. It is also simple to prove that $C(1, u_1, ..., u_{d-1})$ is a $(d-1)$-dimension copula and, more generally that all k-dimensional marginals with $2 \leq k \leq d$ are copulas.We now recall the definition of the quantile function or generalized inverse: for a CDF, F, the generalized inverse, $F^{\leftarrow}$, is defined as

$$F^{\leftarrow}(x) := \inf\{v : F(v) \geq x\}$$

We then have the following well-known result:

Proposition 1. If $U \sim U[0, 1]$ and F_X is a CDF, then

$$P(F^{\leftarrow}(U) \leq x) = F_X(x)$$

In the opposite direction, if X has a continuous CDF, F_X, then

$$F_X(x) \sim U[0, 1]$$

Now let $X = (X_1, ..., X_d)$ be a multivariate random vector with CDF F_X and with continuous and increasing marginals. Then by Proposition 1 it follows that

the joint distribution of $F_{X_1}(X_1), ..., F_{X_d}(X_d)$ is a copula, C_X say. We can find an expression for C_X by noting that

$$\begin{aligned} C_X(u_1, ..., u_d) &= P(F_{X_1}(X_1) \leq u_1, ..., F_{X_d}(X_d) \leq u_d) \\ &= P(X_1 \leq F_{X_1}^{-1}(u_1), ..., X_d \leq F_{X_d}^{-1}(u_d)) \\ &= F_X(F_{X_1}^{-1}(u_1), ..., F_{X_d}^{-1}(u_d)) \end{aligned} \tag{3.1}$$

If we now let $u_j := F_{X_j}(x_j)$ then Eq. **??** yields

$$F_X(x_1, ..., x_d) = C_X(F_{X_1}(X_1), ..., F_{X_d}(X_d))$$

This is one side of the famous Sklar's Theorem which we now state formally

Theorem 1 (Sklar's Theorem 1959). Consider a d-dimensional CDF, F, with marginals $F_1, \ldots, F_d$. Then there exists a copula, C, such that

$$F(x_1, ..., x_d) = C(F_{X_1}(X_1), ..., F_{X_d}(X_d)) \tag{3.2}$$

for all $x_i \in [-\infty, \infty]$ and $i = 1, ..., d$

If F_i is continuous for all $i = 1, ..., d$, then C is unique; otherwise C is uniquely determined only on $Ran(F_1) \times \cdots \times Ran(F_d)$ where $Ran(F_i)$ denotes the range of the CDF, F_i

In the opposite direction, consider a copula, C, and univariate CDF's $F_1, ..., F_D$. Then F as defined Eq. **??** is a multivariate CDF with marginals $F_1, ..., F_D$. □

3.3 Research Direction

The challenges of this research include which copulas are most suitable for capturinge real-world scenarios and determining the architecture for cupula-based adjacency matrix. To address these challenges, we explore different types of bivariate copulas to find the best fit for our data. We then use a graph convolutional network to extract spatial information from the adjacency matrix.

Types of Bivariate Copula

Bivariate Copula is adapt to show the linkage between random variables especially, in financial market **?**. In general, the theory is employed to merge the joint probability distribution functions of two arbitrary marginal distributions. The cumulative distribution functions of two continuous random variables, denoted as X_1 and X_2, are represented by $F_1(x_1)$ and $F_2(x_2)$. From Eq. **??**, a joint bivariate cumulative distribution function $F(x_1, x_2)$ is given by:

$$F(x_1, x_2) = C_\theta \left(u_1 = F_1(x_1), u_2 = F_2(x_2)\right) \tag{3.3}$$

Various copula families that have been suggested are described in **??**. The two most commonly employed types are Elliptical copulas and Archimedean copulas. Archimedean copulas are created from the ground up whereas, Elliptical copulas borrow the correlation structure from existing multivariate distributions. In particular, Clayton, Gumbel, Frank are Archimedean copulas. Whearas, Gaussian is elliptical copula **?**.

Different types of bivariate copula functions exhibit varying tail correlation structures. For instance, the bivariate Gaussian copula demonstrates a symmetric property, limiting its ability to capture asymmetric dependence structures between variables. Conversely, the bivariate Gumbel copula is highly responsive to changes in the upper tail of variable distributions and can quickly capture the upper tailrelated changes. The bivariate Frank copula is well-suited for strong central correlation and weak tail correlation. In contrast, the bivariate Clayton copula can capture lower tail dependence structures in two variables. Consequently, selecting a copula function that better aligns with the data's dependence characteristics results in improved fitting **?**.

$\Phi_\rho(\cdot)$ denotes the standard bivariate normal distribution function whose correlation coefficient matrix is ρ

$\Phi^{-1}(\cdot)$ is the inverse CDF of the standard normal $\Phi(\cdot)$

Name of copula	Bivariate copula	Parameter θ
Gaussian	$\Phi_\rho(\Phi^{-1}(u_1), \Phi^{-1}(u_2))$	-
Clayton	$(u_1^{-\theta} + u_2^{-\theta} - 1)^{-\frac{1}{\theta}}$	$[-1, \infty) \backslash \{0\}$
Gumbel	$\exp\left(-\left((-\ln u_1)^\theta + (-\ln u_2)^\theta\right)^{\frac{1}{\theta}}\right)$	$[1, \infty)$
Frank	$-\frac{1}{\theta}\ln\left(1 + \frac{(e^{-\theta u_1} - 1)(e^{-\theta u_2} - 1)}{e^{-\theta} - 1}\right)$	$\mathbb{R} \backslash \{0\}$

Table 3.1: The characteristics of different bivariate copula functions.

u_1, u_2 denote two arbitrary marginal cumulative distribution functions, with the copula function's dependence parameter ranging between $[-1, 1]$

θ is dependency coefficients

Bivariate Gaussian Copula

The following is a description of the bivariate normal distribution function:

$$C_\rho^{Gauss}(u_1, u_2) = \Phi_\rho(\Phi^{-1}(u_1), \Phi^{-1}(u_2)) \tag{3.4}$$

where:

$\Phi_\rho(\cdot)$ denotes the standard bivariate normal distribution function whose correlation coefficient matrix is ρ

$\Phi^{-1}(\cdot)$ is the inverse CDF of the standard normal $\Phi(\cdot)$

u_1, u_2 denote two arbitrary marginal cumulative distribution functions, with the copula function's dependence parameter ranging between $[-1, 1]$

Due to its symmetrical nature, the Gaussian copula is commonly employed. It is capable of achieving Fréchet lower and upper bounds and capturing a wide range of dependence $(-1, 1)$ between multidimensional variables. However, as the extreme values are approached, the left and right tails tend to zero, making it incapable of capturing asymmetric correlations between variables.

Bivariate Clayton Copula

The generator of the copula is $\varphi(t) = \frac{1}{\theta}(t^{-\theta} - 1)$, the joint cumulative distribution function constructed by Clayton copula can be expressed as:

$$C_\theta^{Cl}(u_1, u_2) = (u_1^{-\theta} + u_2^{-\theta} - 1)^{-\frac{1}{\theta}} \tag{3.5}$$

The Clayton copula exhibits an asymmetrical density function, forming an "L" shape. It is particularly sensitive to left-tail changes and can quickly capture left-tail-related changes. If the dependence structure between two random variables can be characterized by the Clayton copula, it signifies a stronger correlation at the left tail of the distribution. Conversely, at the right tail of the distribution, the variables approach asymptotic independence. As a result, the Clayton copula demonstrates limited sensitivity to changes at the right tail and struggles to capture such variations. The dependency coefficient range for the Clayton copula function lies in $(0, \infty)$. As θ approaches 0, the random variables u_1 and u_2 tend towards independence. On the other hand, as θ tends towards ∞, u_1 and u_2 become completely correlated.

Bivariate Gumbel Copula

As an instance of an Archimedean copula, the copula's generator is given by $\varphi(t) = (-\ln t)^\theta$. The associated Gumbel copula function is defined as follows:

$$C_\theta^{Gu}(u_1, u_2) = \exp\left(-\left((-\ln u_1)^\theta + (-\ln u_2)^\theta\right)^{\frac{1}{\theta}}\right) \tag{3.6}$$

The Gumbel copula exhibits an asymmetrical density function, forming a 'J' shape. It is particularly sensitive to right tail dependence of the variables and can quickly capture the changes associated with right tail dependence. Specifically, if the dependence structure between two random variables can be described by the Gumbel copula, it implies a stronger correlation between the variables at the right

tail. In the right tail of the distribution, where the variables approach asymptotic independence, the Gumbel copula demonstrates limited sensitivity to tail changes and struggles to capture variations in the left tail. The range of the dependency coefficient lies within $(1, \infty)$.

Bivariate Frank Copula

When the generator of the Copula is defined as $\varphi(t) = -\ln\left(\frac{\exp(-\theta t)-1}{\exp(-\theta)-1}\right)$, the resulting Archimedean copula is known as the Frank copula. It can be expressed as follows:

$$C_\theta^{Frank}(u_1, u_2) = -\frac{1}{\theta}\ln\left(1 + \frac{(e^{-\theta u_1}-1)(e^{-\theta u_2}-1)}{e^{-\theta}-1}\right) \tag{3.7}$$

The Frank copula exhibits a symmetrical density function, enabling analysis of both positive and negative dependence structures without limitations on the degree of correlation. It is well-suited for strong central correlation and weak tail correlation scenarios. The range of the dependency parameter extends from $-\infty$ to ∞, while the range of Kendall tau lies within $(-1, 1)$. Consequently, the Frank copula facilitates positive or negative correlation between two variables.

the Graph Convolution Network

From Appendix I, GCN is defined as

$$Z = \hat{A}XW \tag{3.8}$$

where:

$\hat{A} \in \mathbb{R}^{N\times N}$ is the normalized adjacency matrix with self-loop

$X \in \mathbb{R}^{N\times D}$ is the input signal

$W \in \mathbb{R}^{D\times M}$ is the model parameter matrix

$Z \in \mathbb{R}^{N \times M}$ is the output

By leveraging the diffusion convolution layer **?**, which enables the processing of graph signals with K finite steps. We generalize its diffusion convolution layer into the form (**??**), which results in,

$$Z = \sum_{k=0}^{K} P^k X W_k \tag{3.9}$$

where:

$P^k \in \mathbb{R}^{N \times N}$ is the power series of the transition matrix

in the case an undirected graph $P = \frac{A}{\text{rowsum}(A)}$.

in the case an directed graph the diffusion have two directions, forward and backward for the forward transition matrix $P_f = \frac{A}{\text{rowsum}(A)}$, the backward transition matrix $P_b = \frac{A^T}{\text{rowsum}(A^T)}$.

By incorporating both the forward and backward transition matrices, the diffusion graph convolution layer can be expressed as follows:

$$Z = \sum_{k=0}^{K} \left(P_f^k X W_{k1} + P_b^k X W_{k2} \right) \tag{3.10}$$

By integrating predefined spatial dependencies and self-learned hidden graph dependencies (**??**), we introduce the following formulation for the graph convolution layer:

$$Z = \sum_{k=0}^{K} \left(P_f^k X W_{k1} + P_b^k X W_{k2} + \tilde{A}_{adp}^k X W_{k3} \right) \tag{3.11}$$

When the graph structure is unknown and we rely solely on the self-adaptive adjacency matrix to capture hidden spatial connections, the equation can be expressed as follows:

$$Z = \sum_{k=0}^{K} \tilde{A}_{adp}^k X W_k \tag{3.12}$$

Temporal Convolution Layer

To capture the temporal trends of a node, Graph WaveNet use the dilated causal convolution **?** as temporal convolution layer (TCN). By extending the layer depth, dilated causal convolution networks enable an exponentially huge receptive field. Dilated Casual Convolution Networks are able to handle long-range sequences effectively in a non-recursive way, which allows parallel processing and mitigating the gradient explosion problem, in contrast to RNN-based techniques. The dilated causal convolution maintains the temporal causal order by padding the inputs with zeros, ensuring that predictions generated for the current time step only involve historical data. As a specific instance of the standard 1D-Convolution, the dilated causal convolution operation involves sliding over the inputs while selectively skipping values at a certain interval.

Figure 3.1: Dilated casual convolution with kernel size 2. With a dilation factor k.

Given a 1-D sequence input $X \in \mathbb{R}^T$ and a filter $f \in \mathbb{R}^k$, the dilated convolution operation of X with f at step t can be represented as

$$X * f(t) = \sum_{i=0}^{k-1} f(i) \cdot X(t - d \cdot i) \tag{3.13}$$

where:

d is the dilation factor which controls the skipping distance.

k is the size of the filter.

The receptive field of a model increases exponentially when stacked dilated causal convolution layers with increasing numbers of dilation factors as shown in figure (**??**). This conserves processing resources by allowing dilated causal convolution networks to capture longer sequences with fewer layers.

Gated TCN: Gating mechanisms are critical in recurrent neural networks. It has been demonstrated that they are effective in controlling how information moves between layers in temporal convolution networks as well **?**. To understand complex temporal connections, WaveNet uses Gated TCN in the model. A simple Gated TCN only contains an output gate. Given the input $X \in \mathbb{R}^{N \times D \times S}$, it takes the form

$$\mathbf{h} = g(\Theta_1 * X + b) \odot \sigma(\Theta_1 * X + c) \tag{3.14}$$

where:

Θ_1, Θ_2, b, c are model parameters.

$\odot$ is the element-wise product.

g is an activation function of the outputs.

σ is sigmoid function which determines the ratio of information passed to the next layer.

Chapter IV

RESEARCH METHODS

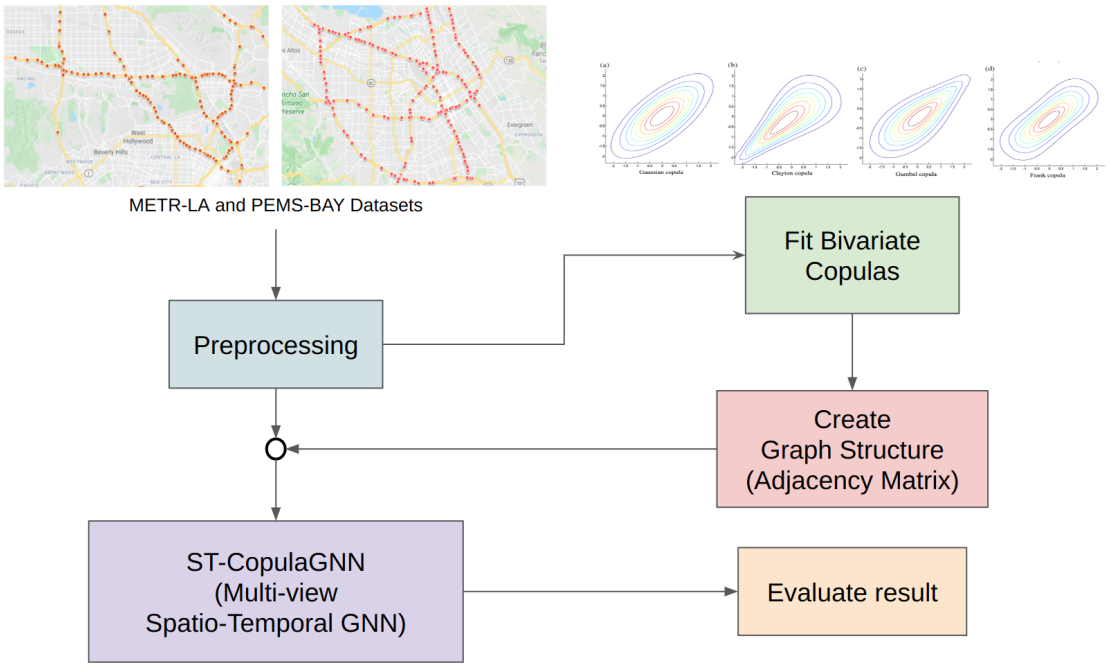


Figure 4.1: Flowchart of the framework.

The first step involves preprocessing the raw data and identifying the best-fitted bivariate copulas for each node pair. The next step is to generate an adjacency matrix for each family of copulas. The preprocessed data and adjacency matrices are then used to train a multi-view spatio-temporal graph neural network model. Finally, the performance of this model is evaluated and compared against previous methods.

4.1 Fit Copula to Traffic Dataset

In particular, Clayton, Gumbel, Frank are Archimedean copulas. Whereas, Gaussian is elliptical copulas. Bivariate copulas tend to need just 1 parameter to specify the shape. In particular Clayton, Gumbel, Frank, Gaussian are 1-parameter copulas, whereby $\theta_{Clayton} \in [-1, \infty) \backslash \{0\}, \theta_{Gumbel} \in [1, \infty), \theta_{Frank} \in \mathbb{R} \backslash \{0\}, \rho_{Gaussian} \in$

$[-1, 1]$.

Bivariate copulas can be thought of as a surface over a square grid, with $u \in [0, 1]$ value on the horizontal axis and $v \in [0, 1]$ value on the vertical axis). From there, one can distinguish between unidirectional copulas that only depict positive dependence (generalised from notion of positive Pearson correlation) vs. bidirectional copulas that can depict negative dependence (generalised from notion of negative Pearson correlation) as well. For the unidirectional copulas, negative dependence can be accommodated by "flipping" the u value, i.e. working with $(-u, v)$ instead of the original (u, v) coordinate pair, otherwise designated as making a $90°$ rotation.

One can also distinguish between symmetric copulas (upside pattern same as downside) vs. asymmetric copulas (upside pattern different from downside). For the asymmetric copulas, different result (better or worse fit) may be achieved by "flipping" both the u and the v values, i.e. working with $(-u, -v)$ instead of the original (u, v) coordinate pair, otherwise designated as making a $180°$ rotation.

Finally for asymmetric and unidirectional copulas, negative dependence can also be accommodated by "flipping" the v value, i.e. working with $(u, -v)$ instead of the original (u, v) coordinate pair, otherwise designated as making a $270°$ rotation, achieving different result (better or worse fit) as compared with making the $90°$ rotation. Note also that $180°$ rotation version of a copula is also/often referred to as survival copula.

In order to fit the copulas, we must first specify the univariate (marginal) distribution for each of our n random variables in order to turn data samples into ones of uniform distribution. Here, the "cleanest" approach is to use the (inverse of) empirical cdf throughout. Effectively, the original n time series of each detector become "uniformised", each resembling a set of Uniform$(0, 1)$ random variates. We are interested in analysing pair-wise linkages, of which there are $\frac{n(n-1)}{2}$ unique pairings.

In Figure **??**, show empirical cdf between node 80 and node 195 on the top left then fitted with bivariate copulas (Gaussian, Clayton, Gumbel, Frank) and the best fitted is Clayton. Then we collect copula parameter θ to create adjacency matrix.

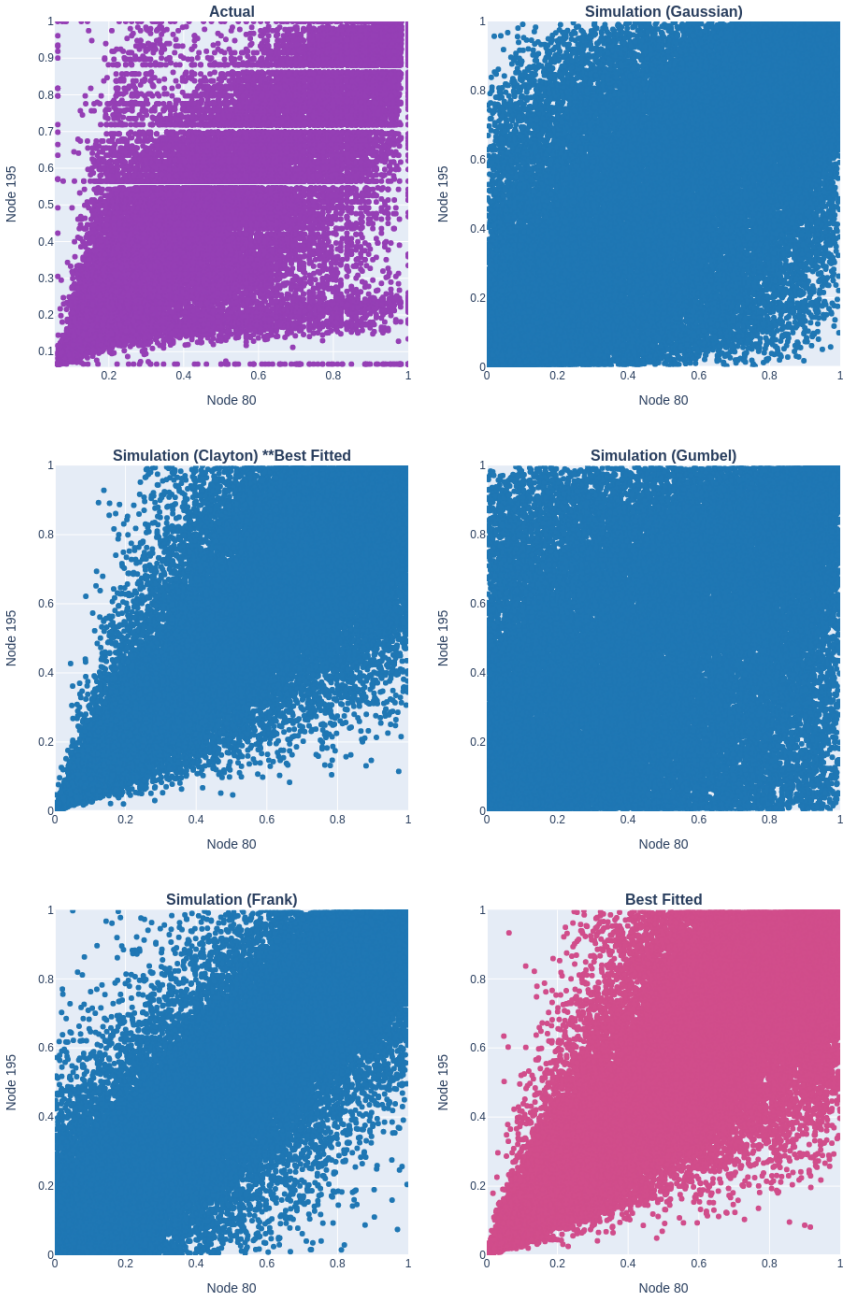


Figure 4.2: Fit copula to nodes 80 and 195 from METR-LA dataset and simulate to find the best fit.

4.2 Model Architecture

The ST-CopulaGNN model uses the Graph WaveNet framework as a stacked spatio-temporal layers, as depicted in Figure **??**. The architecture has been modified by increasing the size of certain internal convolutional layers and adding a skip connection. To extract data in both realistic and statistical domains, our model consists of two parallel stacked spatial-temporal layers, as shown in Figure **??**. Each spatial-temporal layer, illustrated in Figure **??**, comprises a graph convolution layer (GCN) and a gated temporal convolution layer (Gated TCN) that includes two parallel temporal convolution layers (TCN-a and TCN-b).

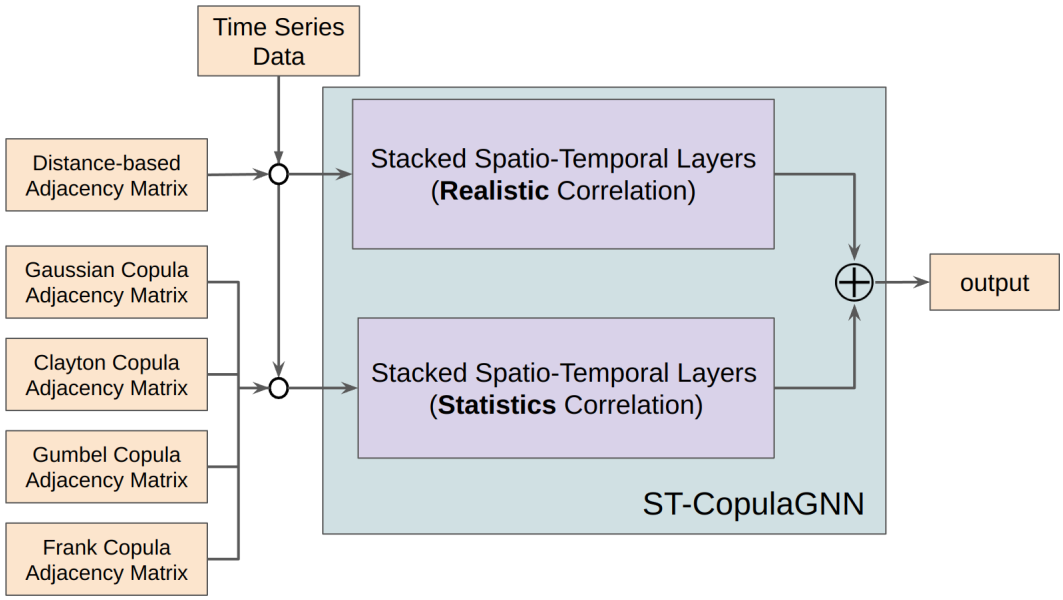


Figure 4.3: Multi-view model that can capture data from both the realistic and statistical domains.

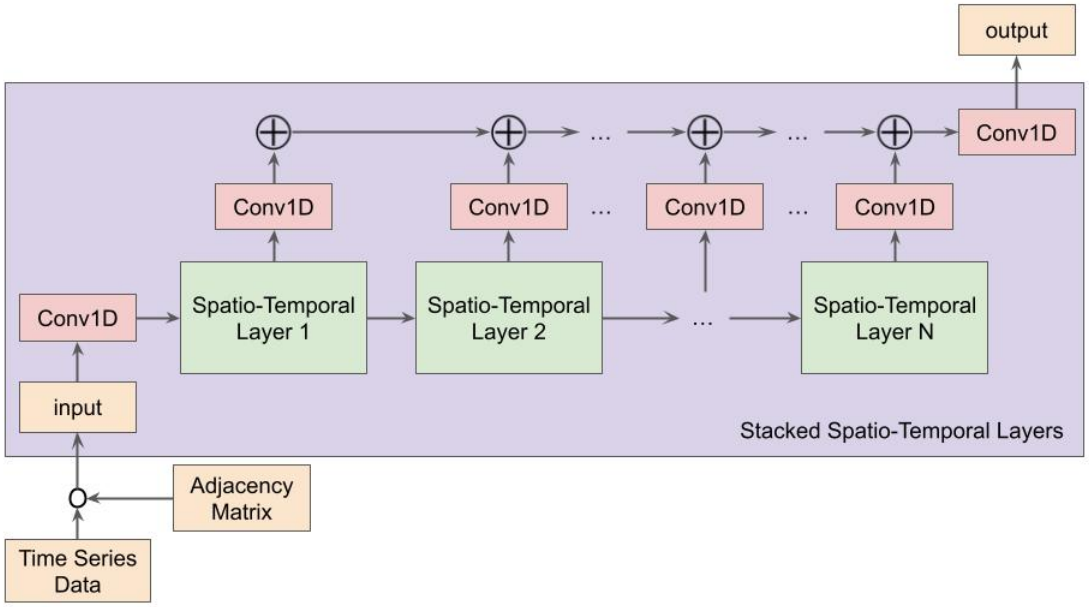


Figure 4.4: Stacked Spatio-Temporal layers.

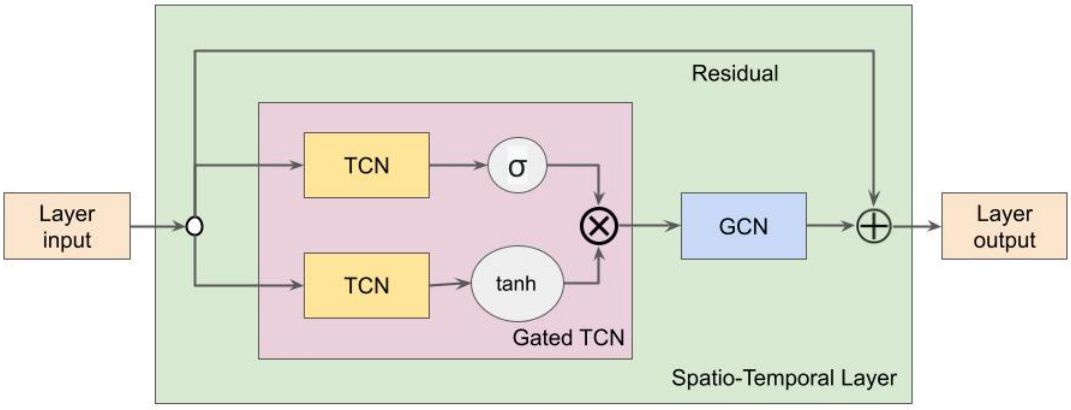


Figure 4.5: Spatio-Temporal layer.

4.3 Experiment Setting

Baselines

We compare our model with the following models.

- ARIMA: Auto-Regressive Integrated Moving Average model with Kalman filter.
- SVR: Support Vector Regression. It is a variant method of the support vector machine model. In this thesis, we use the linear kernel function for traffic prediction.
- FNN: Feedforward Neural Network is a simple neural network model containing two hidden layers and L2 regularization.
- FCN-LSTM: FCN-LSTM applies LSTM to extract temporal in-formation, and outputs final prediction using two full-connected layers.
- DCRNN: Diffusion Convolutional Recurrent Neural Network combines recurrent neural networks with diffusion convolution, modeling both inflow and outflow relationships.
- Graph WaveNet: Graph WaveNet is a convolution network architecture, which introduces a self-adaptive graph to capture the hidden spatial dependency, and uses dilated convolution to capture the temporal dependency.

Datasets

Our study employs the traffic dataset that is commonly used in related works such as Graph WaveNet and DCRNN.

The dataset used in our study, METR-LA, is collected from sensors installed along the highways in Los Angeles County. These sensors capture the velocity of passing vehicles, which are then averaged over 5-minute intervals. Our experiments

Table 4.1: Summary statistics of datasets.

Dataset	Nodes	Edges	Time Range
METR-LA	207	1515	2012/03/01 - 2012/06/27
PEMS-BAY	325	2369	2017/01/01 - 2017/06/30

were conducted on the METR-LA dataset, and we also evaluated the performance of our modifications on a larger dataset, PEMS-BAY, which consists of 6 months of data from the Bay Area and have a similar structure to METR-LA. For the information of the datasets shown in Table **??** and the geographical location for METR-LA and PEMS-BAY shown in Figure **??** and Figure **??** respectively.

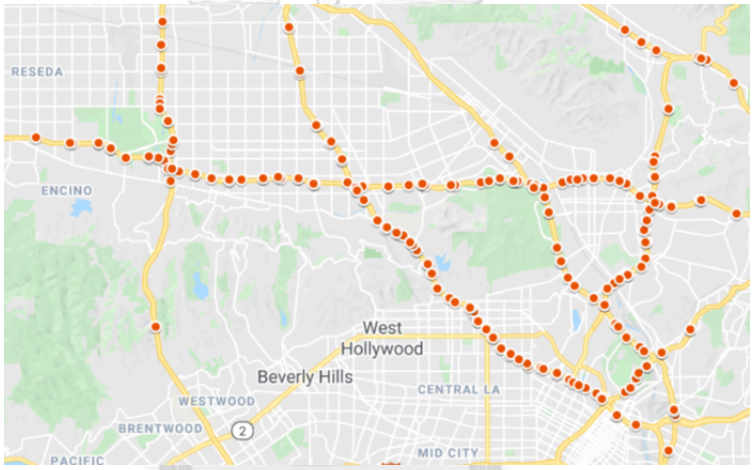


Figure 4.6: Map showing the distribution and location of sensors in METR-LA dataset.

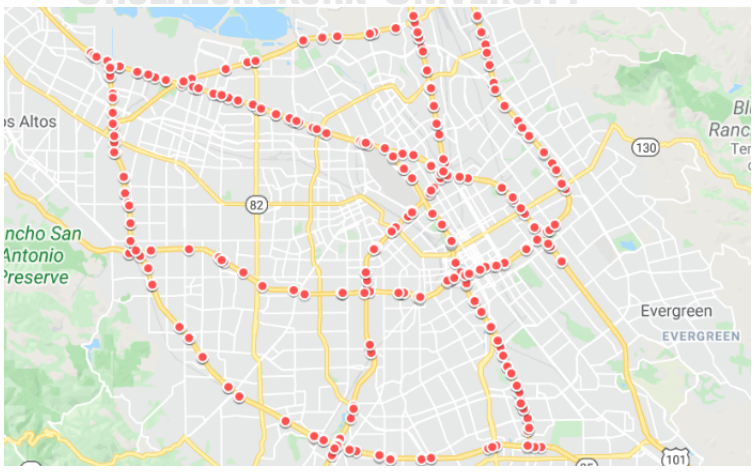


Figure 4.7: Map showing the distribution and location of sensors in PEMS-BAY dataset.

Evaluation Metrics

Three commonly used metrics in traffic forecasting are used in this thesis, including

1. Mean Absolute Error (MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{4.1}$$

2. Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|^2} \tag{4.2}$$

3. Mean Absolute Percentage Error (MAPE)

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \tag{4.3}$$

where:

y_i is actual target value.

$\hat{y}_i$ is prediction value.

n is number of datapoints.

Chapter V

RESULTS

5.1 Experiment Results

Similar to DCRNN and Graph WaveNet, we divided the data into three sets for training, validation, and testing, with a ratio of 70%, 10%, and 20% respectively. The division was based on the chronological order of the data, where the training set came before the validation set, which in turn preceded the test set. For model selection and early stopping, we used the MAE on the validation set, but we also reported the MAE on the test set.

Table 5.1: Comparing the performance of ST-CopulaGNN against other baseline models.

	15 minutes			30 minutes			1 hour		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
METR-LA									
ARIMA	4.01	9.64%	8.22	5.15	12.72%	10.47	6.91	17.40%	13.20
Support Vector Regression	3.52	8.79%	7.94	4.46	11.61%	10.04	5.64	15.88%	12.02
Feedforward Neural Network	3.85	9.49%	7.72	4.93	12.65%	9.65	6.44	17.06%	11.39
FCN-LSTM	3.49	9.60%	6.32	3.81	10.97%	7.37	4.45	13.72%	8.93
DCRNN (Reported)	2.77	7.30%	5.38	3.15	8.80%	6.45	3.60	10.50%	7.60
Graph WaveNet	2.43	6.02%	4.44	2.68	7.00%	5.14	3.03	8.18%	6.07
ST-CopulaGNN	**2.39**	**5.73%**	**4.40**	**2.65**	**6.70%**	5.14	**3.02**	**7.88%**	6.07
PEMS-BAY									
ARIMA	1.65	3.55%	3.40	2.32	5.41%	4.94	3.38	8.30%	6.50
Support Vector Regression	1.53	3.35%	3.35	2.08	4.89%	4.82	2.73	7.06%	6.26
Feedforward Neural Network	1.54	3.27%	3.13	2.10	4.81%	4.43	2.73	6.55%	5.60
FCN-LSTM	2.11	4.86%	4.32	2.25	5.31%	4.62	2.47	5.93%	5.02
DCRNN (Reported)	1.38	2.90%	2.95	1.74	3.90%	3.97	2.07	4.90%	4.74
Graph WaveNet	**1.13**	2.33%	2.23	1.39	**3.02%**	2.93	1.66	3.90%	3.68
ST-CopulaGNN	1.14	**2.32%**	2.23	**1.38**	3.04%	**2.87**	**1.65**	**3.81%**	**3.64**

Table **??** compares the performance of ST-CopulaGNN and baseline models for 15 minutes, 30 minutes and 1 hour ahead prediction on METR-LA and PEMS-BAY datasets. In comparison to other models, ARIMA and SVR (Support Vector Regression) perform decently but have higher MAE, MAPE, and RMSE values than ST-CopulaGNN. The Feedforward Neural Network and FCN-LSTM models also have higher MAE, MAPE, and RMSE values compared to ST-CopulaGNN for all prediction ranges. When compared to other spatial-temporal models, ST-CopulaGNN performed better than the previous recurrent-based approach DCRNN

and the graph neural network approach Graph WaveNet with a slight improvement.

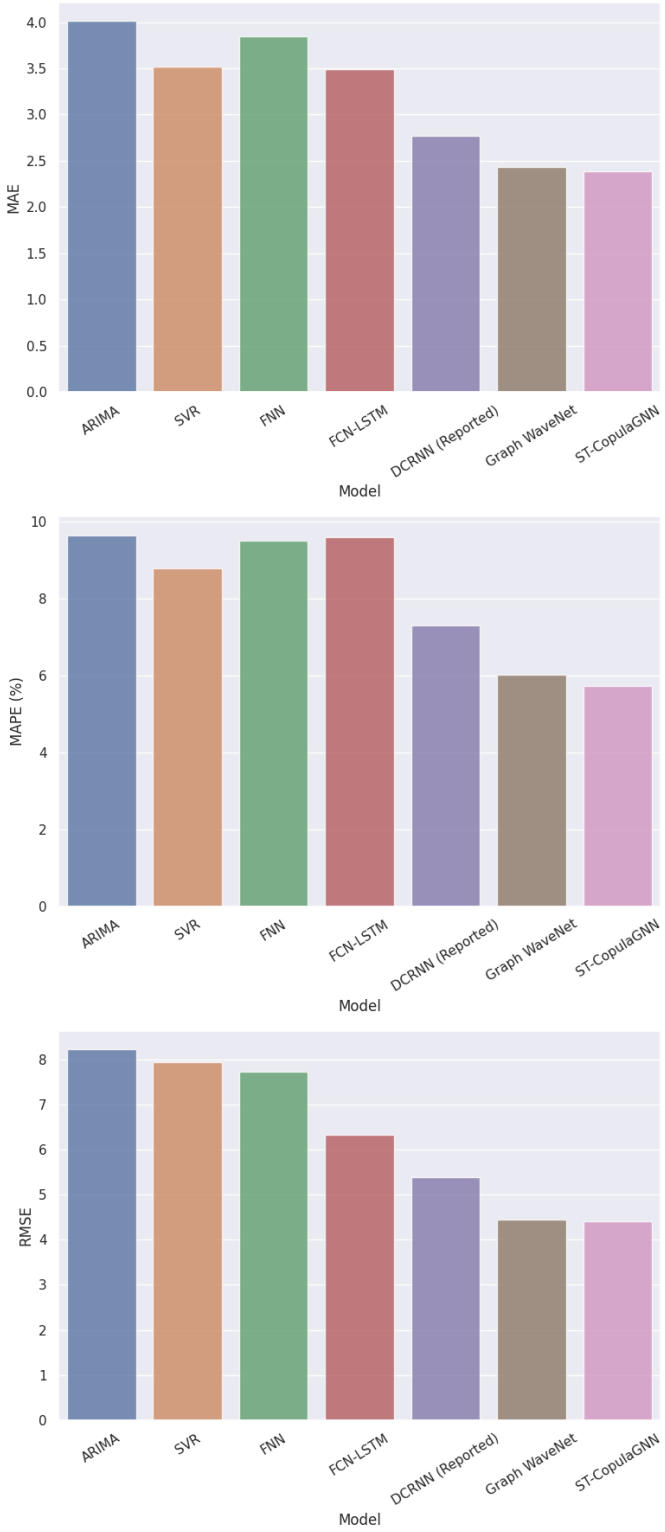


Figure 5.1: Comparing the performance of ST-CopulaGNN against other baseline models on METR-LA dataset for 15 minutes ahead prediction.

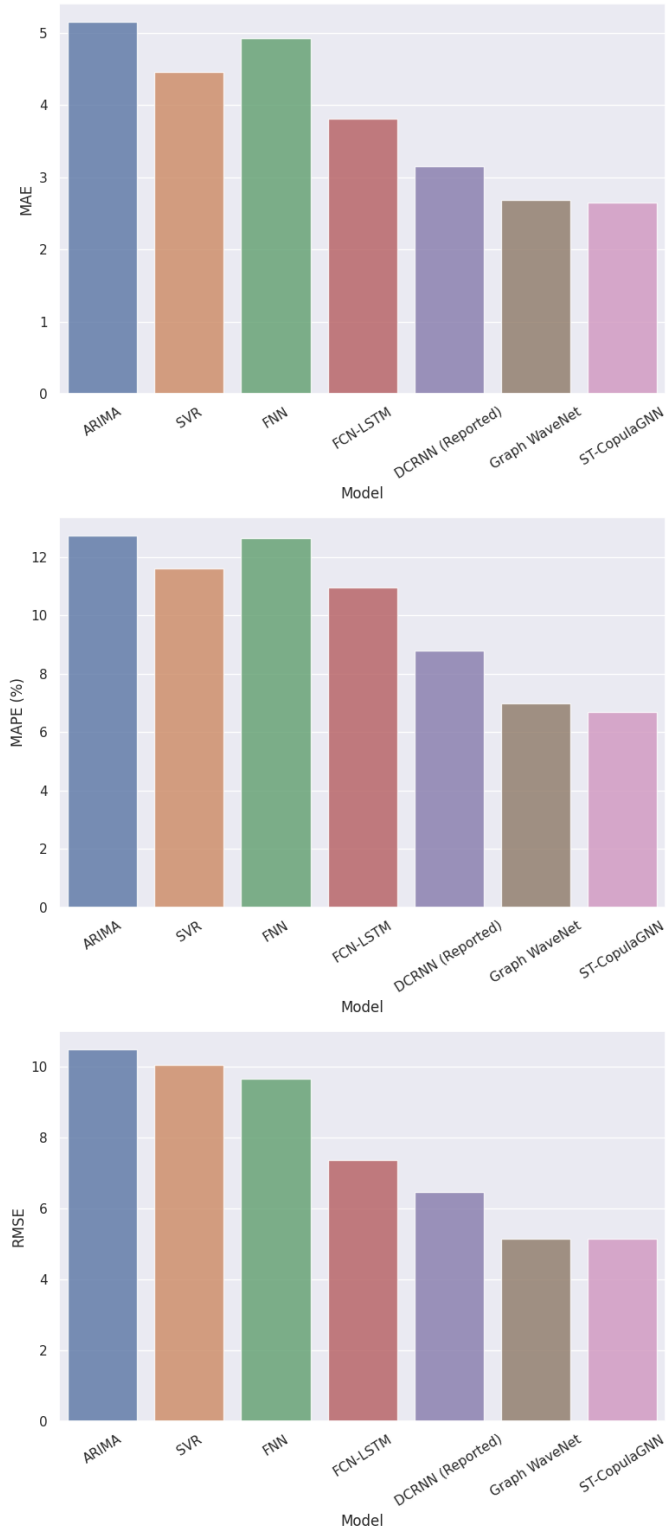


Figure 5.2: Comparing the performance of ST-CopulaGNN against other baseline models on METR-LA dataset for 30 minutes ahead prediction.

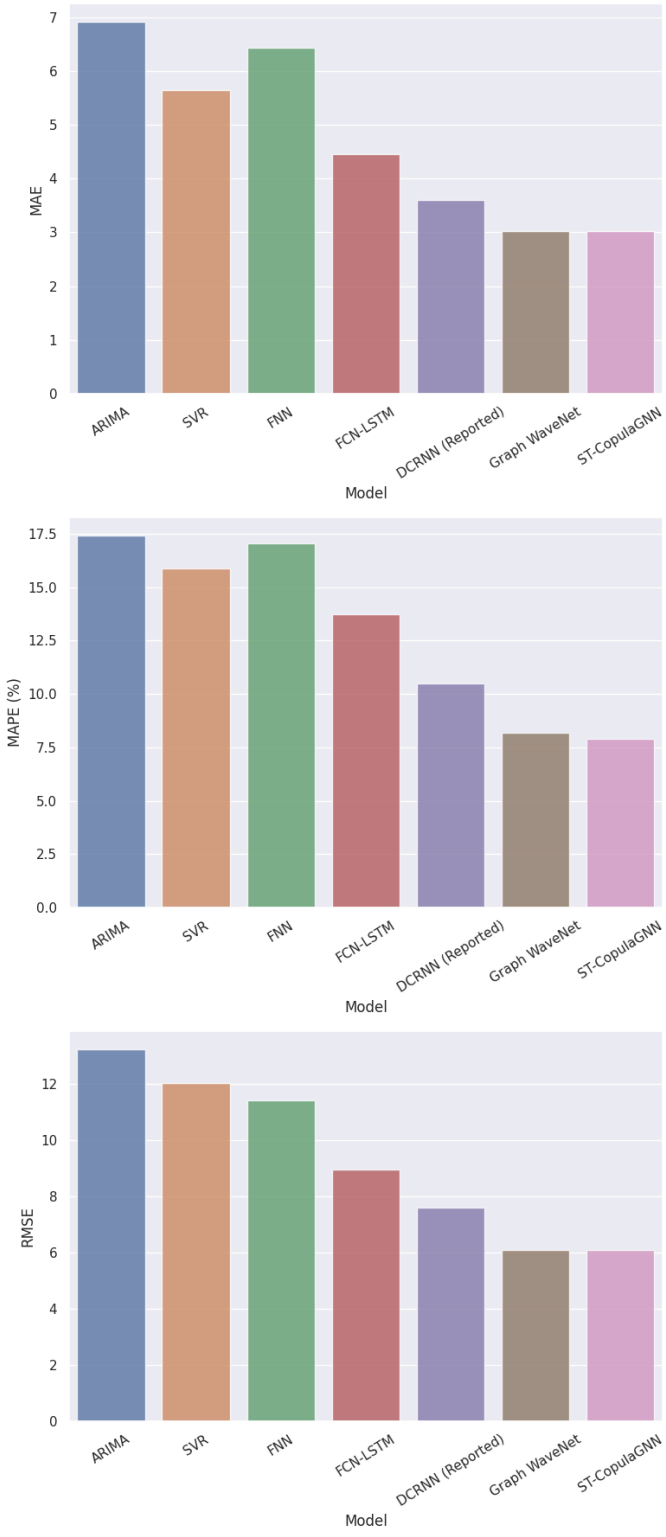


Figure 5.3: Comparing the performance of ST-CopulaGNN against other baseline models on METR-LA dataset for 1 hour ahead prediction.

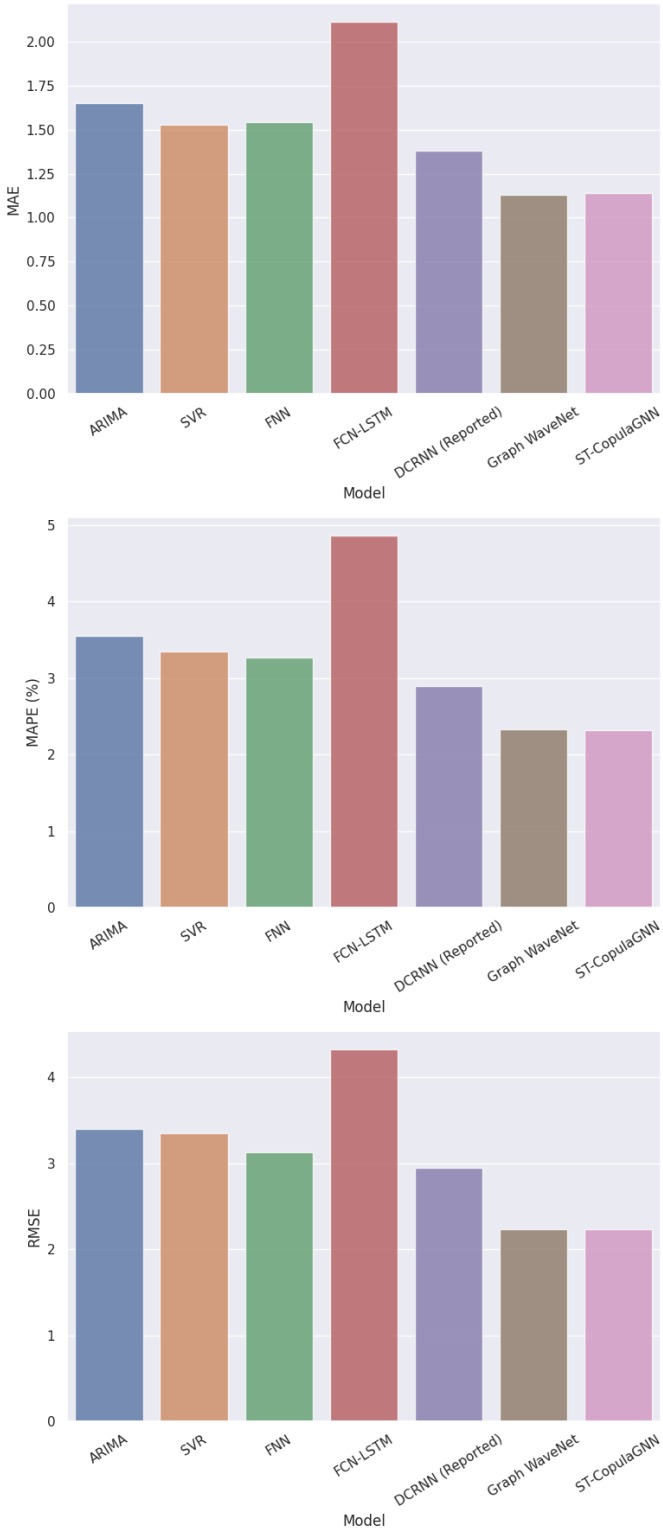


Figure 5.4: Comparing the performance of ST-CopulaGNN against other baseline models on PEMS-BAY dataset for 15 minutes ahead prediction.

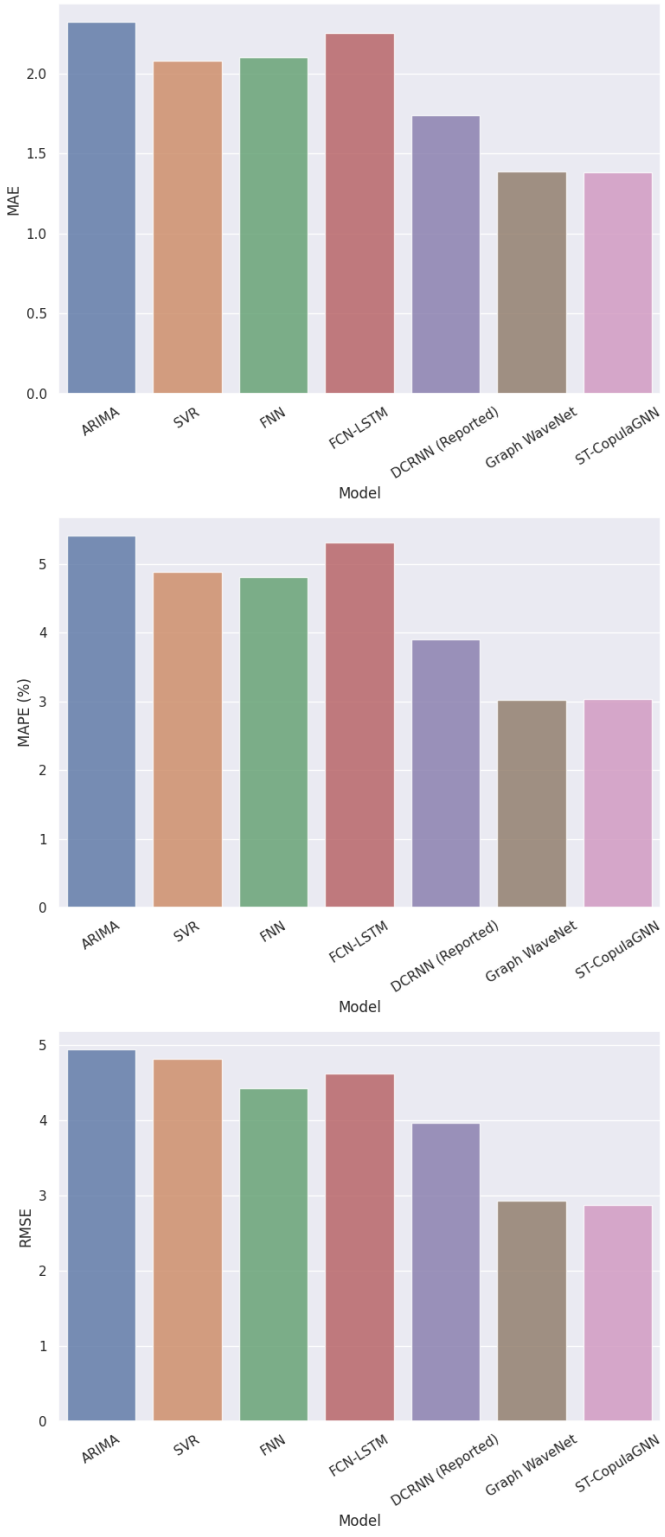


Figure 5.5: Comparing the performance of ST-CopulaGNN against other baseline models on PEMS-BAY dataset for 30 minutes ahead prediction.

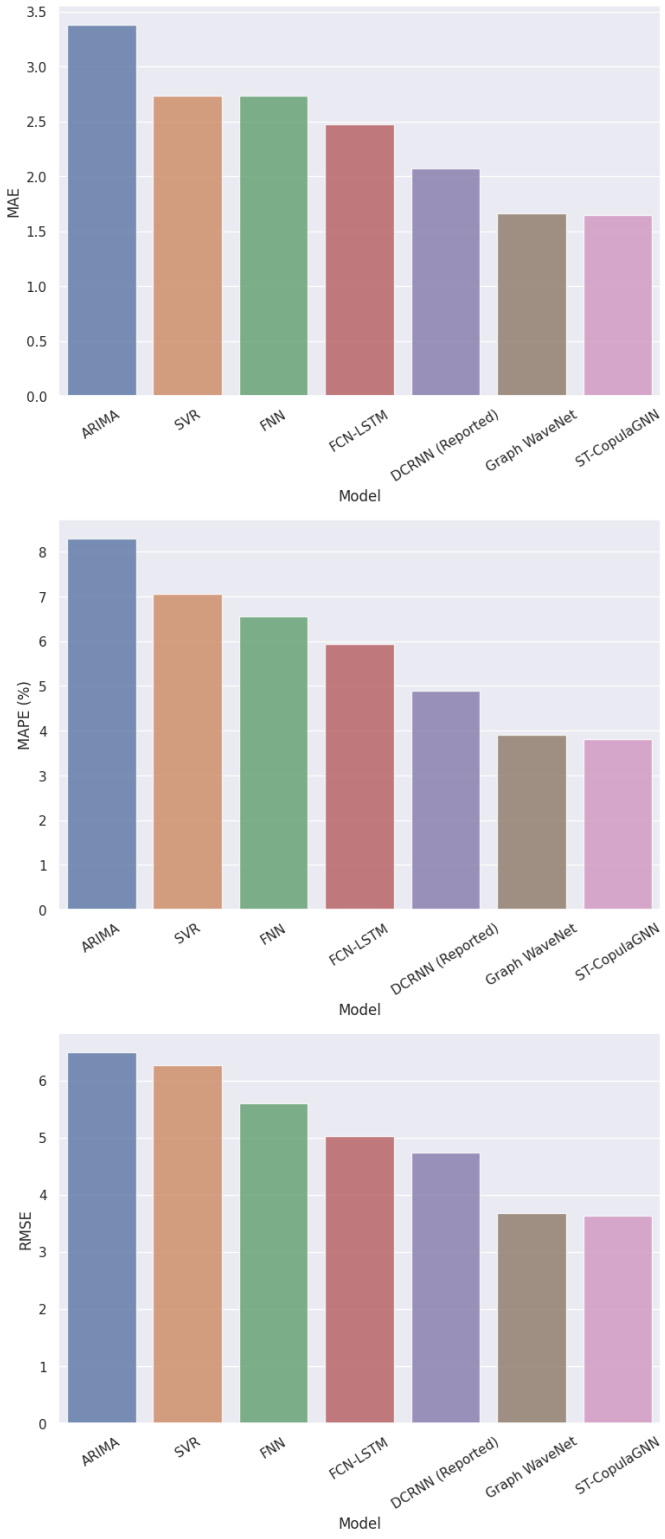


Figure 5.6: Comparing the performance of ST-CopulaGNN against other baseline models on PEMS-BAY dataset for 1 hour ahead prediction.

Starting with the METR-LA dataset, for 15 minutes ahead prediction, ST-CopulaGNN outperforms all other models, including Graph WaveNet, which was the previous best performer. ST-CopulaGNN achieves an MAE of 2.39, which is better than Graph WaveNet's MAE of 2.43. Similarly, for 30 minutes ahead prediction, our model outperforms all other models, including Graph WaveNet, with an MAE of 2.65, which is better than Graph WaveNet's MAE of 2.68. For 1 hour ahead prediction, ST-CopulaGNN is again the best performer with an MAE of 3.02, which is slightly lower than the second-best performing model Graph WaveNet with an MAE of 3.03. For the PEMS-BAY dataset, ST-CopulaGNN again achieves the lowest MAE of 1.38 and 1.65 for 30 minutes and 1 hour ahead predictions respectively, which is lower than the previously DCRNN and Graph WaveNet models. The model also achieves the lowest MAPE of 2.32% and an RMSE of 2.23 for 15 minutes ahead prediction, which are both lower than the previously reported models. Overall, the results suggest that the proposed ST-CopulaGNN model is a promising method for spatio-temporal traffic forecasting, outperforming several existing models in terms of accuracy metrics.

This is due to our architecture being designed to effectively detect spatial dependencies across different domains, including realistic and statistical correlations. Figure **??** displays a comparison of 1-hour-ahead predicted values versus real values of ST-CopulaGNN and Graph WaveNet on a snapshot of the test data. The results show that ST-CopulaGNN provides more stable predictions and does not overestimate the values.

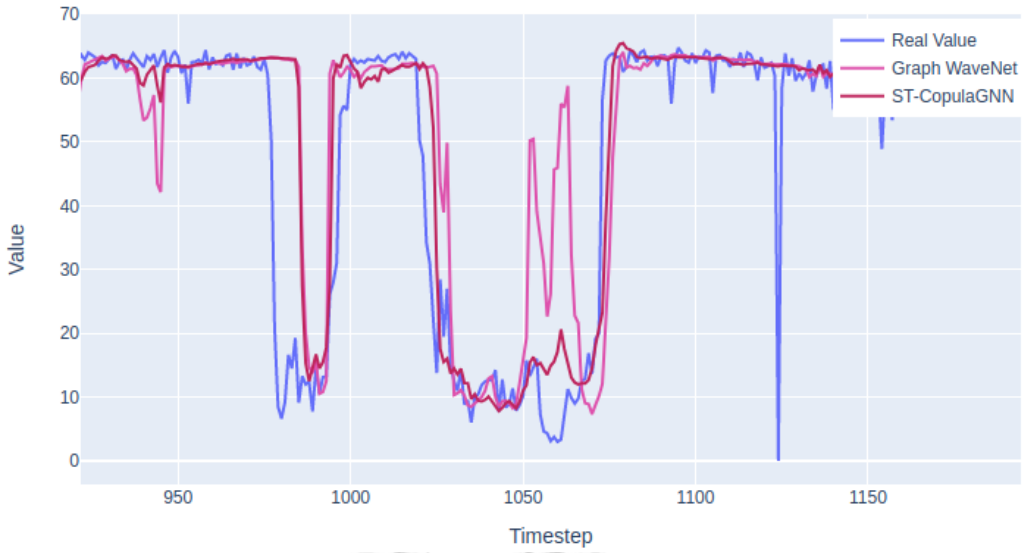


Figure 5.7: Comparison of the prediction curves for a 1-hour-ahead prediction on a snapshot of the METR-LA test data, between ST-CopulaGNN and Graph WaveNet.

5.2 Effect of Different Adjacency Matrices

Table 5.2: Experimental results of different adjacency matrices (with and without time lag) configurations on METR-LA dataset for one hour ahead prediction.

	w/o time lag			w/ time lag		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE
Gaussian	3.016	7.9%	6.047	3.016	8.0%	6.096
Clayton	**2.988**	7.8%	**5.979**	2.999	7.8%	6.037
Gumbel	3.019	7.8%	6.055	3.022	7.9%	6.062
Frank	2.992	7.8%	6.019	3.030	8.0%	6.083
Gaussian, Clayton	3.028	7.9%	6.066	3.017	8.0%	6.087
Gaussian, Gumbel	3.000	8.0%	6.055	3.016	8.0%	6.099
Gaussian, Frank	3.001	8.0%	6.052	3.000	7.9%	5.991
Clayton, Gumbel	3.019	7.9%	6.067	**2.984**	7.8%	5.999
Clayton, Frank	3.016	7.9%	6.085	2.997	7.8%	6.054
Gumbel, Frank	3.047	7.9%	6.158	2.988	7.8%	6.024
Gaussian, Clayton, Gumbel	2.992	7.9%	6.014	3.000	7.9%	6.020
Gaussian, Clayton, Frank	3.027	7.9%	6.091	3.004	7.9%	6.032
Gaussian, Gumbel, Frank	3.018	8.0%	6.122	2.989	7.9%	**5.981**
Clayton, Gumbel, Frank	3.001	7.9%	6.030	3.019	7.9%	6.067
Gaussian, Clayton, Gumbel, Frank	3.018	7.9%	6.069	2.987	7.8%	6.010

The performance of our proposed ST-CopulaGNN model was evaluated using different copula adjacency matrices for one hour ahead prediction on METR-LA dataset. The results are presented in Table **??**, which summarizes the mean absolute error (MAE), mean absolute percentage error (MAPE), and root mean squared error

(RMSE) for each copula configuration.

Overall, the results indicate that the choice of copula function has a significant impact on the performance of the model. Among the individual copulas, the Clayton copula achieved the best results in terms of both MAE (2.988) and RMSE (5.979). The Frank copula also performed well, achieving an MAE of 2.992 and an RMSE of 6.019. When using multiple copulas, the combination of Gaussian and Clayton copulas resulted in an MAE of 3.028 and an RMSE of 6.066, which is slightly worse than using the Clayton copula alone. The combination of Gaussian and Frank copulas also performed worse than using the Frank copula alone, with an MAE of 3.001 and an RMSE of 6.052. The combination of all four copulas (Gaussian, Clayton, Gumbel, and Frank) resulted in an MAE of 3.018 and an RMSE of 6.069, which is comparable to using the Clayton copula alone. In summary, the results suggest that the Clayton copula is the most effective for our ST-CopulaGNN model when used on METR-LA dataset. However, using multiple copulas can also provide good results, with the combination of all four copulas performing comparably to using the Clayton copula alone.

In addition, by staggering paired time-series with time lag on one of the pairs, the resulting copula relationship is suggestive of information flow. The results show that the (Clayton, Gumbel), and (Gaussian, Gumbel, Frank) adjacency matrix configurations outperform the other configurations, with the lowest MAE and RMSE values of 2.984 and 5.999, and 2.989 and 5.981, respectively. These results suggest that using copula-based adjacency matrix configurations, particularly those that include Clayton and Gumbel copulas, can improve traffic forecasting accuracy. Additionally, the use of paired time-series with time lag may reveal information flow relationships between the nodes in the network.

Chapter VI

DISCUSSION

6.1 Conclusion

This thesis discusses the application of statistical terms in graph neural networks. We present a multi-view spatio-temporal model capable of capturing both the realistic and statistical domains. By combining graph convolution with dilated causal convolution, our model effectively and efficiently captures spatial-temporal dependencies. We also propose an effective method to measure linkage correlation between pair of nodes involves utilizing Copula, which provides a modern approach with a rigid structure capable of describing not only the relationship between random variables, but also any other properties relevant to the entire structure. We conducted experiments to determine the best fit copula, both with and without time lag, and found that the optimal copula varied across time in the METR-LA dataset. Specifically, we discovered that the Clayton Copula performed best without time lag. In contrast, with time lag, the combination of Clayton and Gumbel copulas produced the best results.

6.2 Future Work

In our future research, we aim to investigate scalable techniques for implementing ST-CopulaGNN on large datasets. Additionally, we plan to explore other statistical approaches, such as Granger causality and different Copula families, to define the linkage between traffic nodes.

Appendix I

SPECTRAL GRAPH CONVOLUTION

The history of Spectral Graph Convolution is described in **??**.

A.1 Overview

In the Fourier domain, the convolution operator on graph $\cdot_G$ is defined as

$$g\left(\cdot_G\right) f = \mathcal{F}^{-1}\left(\mathcal{F}(g) \odot \mathcal{F}(f)\right) = U\left(U^T g \odot U^T f\right) = U g_\theta(\Lambda) U^T f = g_\theta(\mathcal{L}) f \tag{A.1}$$

where $(\cdot_G)$ is convolution operator defined on graph, $\odot$ is Hadamard product. It follows that a signal f is filtered by $g \in \mathbb{R}^n$, and denotes $g_\theta(\Lambda) = diag(U^T g)$ which the diagonal corresponds to spectral filter coefficients.

For details,

$$\begin{aligned}
g_\theta\left(\cdot_G\right) f &= g_\theta(\mathcal{L}) f = g_\theta(U \Lambda U^T) f = U g_\theta(\Lambda) U^T f \\
&= U \begin{bmatrix} \hat{g}(\lambda_1) & & & \\ & \hat{g}(\lambda_2) & & \\ & & \ddots & \\ & & & \hat{g}(\lambda_n) \end{bmatrix} U^T f \\
&= U \begin{bmatrix} \hat{g}(\lambda_1) & & & \\ & \hat{g}(\lambda_2) & & \\ & & \ddots & \\ & & & \hat{g}(\lambda_n) \end{bmatrix} \hat{f} \\
&= U \begin{bmatrix} \hat{g}(\lambda_1) & & & \\ & \hat{g}(\lambda_2) & & \\ & & \ddots & \\ & & & \hat{g}(\lambda_n) \end{bmatrix} \begin{bmatrix} \hat{f}(\lambda_1) \\ \hat{f}(\lambda_2) \\ \vdots \\ \hat{f}(\lambda_n) \end{bmatrix}
\end{aligned} \tag{A.2}$$

$$= \begin{bmatrix} \hat{g}(\lambda_1) \\ \hat{g}(\lambda_2) \\ \vdots \\ \hat{g}(\lambda_n) \end{bmatrix} \odot \begin{bmatrix} \hat{f}(\lambda_1) \\ \hat{f}(\lambda_2) \\ \vdots \\ \hat{f}(\lambda_n) \end{bmatrix} \tag{A.3}$$

Spectral-based GCN all follow this definition of $Ug_\theta(\Lambda)U^T f$, the main difference between different version of Spectral-based GCN lies in the choice of the filter $g_\theta(\Lambda)$ **?**.

A.2 Spectral CNN

Bruna et al. propose the first spectral convolutional neural network **?**. A graph can be associated with node signal $f \in \mathbb{R}^{n \times C_k}$ is a feature matrix with $f_i \in \mathbb{R}^{C_k}$ representing the feature vector of node i. A construction where each layer $k = 1, 2, \ldots, K$ transforms an input vector f^k of size $n \times C_k$ into an output f^{k+1} of size $n \times C_{k+1}$.

$$f_j^{(k+1)} = \sigma\left(U \sum_{n=1}^{C_k} g_{\theta_{i,j}^{(k)}} U^T f_i^{(k)}\right) = \sigma\left(U \sum_{n=1}^{C_k} g_{\theta_{i,j}^{(k)}} \hat{f}_i^{(k)}\right) \tag{A.4}$$

where $g_{\theta_{i,j}^{(k)}}, i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, C_k$ is a diagonal matrix with trainable parameters $\theta_m^{(k)}, m \in (1, n), \sigma$ is activation function. $g_{\theta_{i,j}^{(k)}}$ is given by

$$g_{\theta_{i,j}^{(k)}} = \begin{bmatrix} \theta_1^{(k)} & & & \\ & \theta_2^{(k)} & & \\ & & \ddots & \\ & & & \theta_n^{(k)} \end{bmatrix} \tag{A.5}$$

A.3 ChebNet

ChebNet **?** uses Chebyshev polynomials instead of convolutions in spectral domain. Furthermore, it was demonstrated that that $g_\theta(\Lambda)$ can be approximated by

a truncated expansion in terms of Chebyshev polynomials.

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), n \in \mathbb{N}^+, \tag{A.6}$$

where $T_0(x) = 1, T_1(x) = x$. Here, we make $\tilde{\Lambda} = \frac{2\Lambda}{\lambda_{max}} - I_n \in [-1, 1]$, λ_{max} is the biggest eigenvalue from $\mathcal{L}$

$$g_\theta(\Lambda) = \sum_{k=1}^{K-1} \theta_k T_k(\tilde{\Lambda}) \tag{A.7}$$

where the parameter $\theta \in \mathbb{R}^K, T_k(\tilde{\Lambda}) \in \mathbb{R}^{n \times n}$. The filtering operator can also be written as

$$g_\theta(\mathcal{L})f = \sum_{k=1}^{K-1} \theta_k T_k(\tilde{\mathcal{L}})f \tag{A.8}$$

where $T_k(\tilde{\mathcal{L}}) \in \mathbb{R}^{n \times n}$ is the Chebyshev polinomial of order k evaluated at the scaled Laplacian $\tilde{\mathcal{L}} = \frac{2\mathcal{L}}{\lambda_{max}} - I_n$. Accordingly, spectral filters represented by K^{th} h-order polynomials of the Laplacian are exactly K-localized, i.e. it depends only on nodes that are at maximum K steps away from the central node.

Lemma. Let G be a weighted graph, with adjacency matrix A. Let B equal the adjacency matrix of the binarized graph, i.e. $B_{m,n} = 0$ if $A_{m,n} = 0$ and $B_{m,n} = 1$ if $A_{m,n} > 0$. Let $\tilde{B}$ be the adjacency matrix with unit loops added on every vertex, e.g. $\tilde{B}_{m,n} = B_{m,n}$ for $m \neq n$ and $\tilde{B}_{m,n} = 1$ for $m = n$

Then for each $s > 0, (B^s)_{m,n}$ equals the number of paths of length s connecting m and n, and $(\tilde{B}^s)_{m,n}$ equals the numebr of all paths of length $r \leq s$ connecting m and n.

The Lemma can be used to demonstrate that matrix elements of low powers of the graph Laplacian corresponding to sufficiently separated vertices must be zero. Therefore, **dist**$(v_i, v_j) > K$ implies $(\mathcal{L}^K)_{i,j} = 0$, and the spectral filters of ChebNet are exactly K-localized.

Accordingly,

$$
\begin{aligned}
g_\theta(\Lambda) &= \begin{bmatrix} \hat{g}(\lambda_1) & & & \\ & \hat{g}(\lambda_2) & & \\ & & \ddots & \\ & & & \hat{g}(\lambda_n) \end{bmatrix} \\
&= \begin{bmatrix} \sum_{k=1}^{K-1} \theta_k T_k(\hat{\lambda}_1) & & & \\ & \sum_{k=1}^{K-1} \theta_k T_k(\hat{\lambda}_2) & & \\ & & \ddots & \\ & & & \sum_{k=1}^{K-1} \theta_k T_k(\hat{\lambda}_n) \end{bmatrix}
\end{aligned}
\tag{A.9}
$$

where θ_k is a vector of Chebyshev coefficients, which is trainable parameter. Furthermore, Eq. **??** can be deduced as following

$$
\begin{aligned}
f\left(\cdot_G\right) g_\theta &= g_\theta(U\Lambda U^T)f = U\sum_{k=1}^{K-1}\theta_k T_k(\tilde{\Lambda})U^T f = \sum_{k=1}^{K-1} U\theta_k T_k(\tilde{\Lambda})U^T f \\
&= \sum_{k=1}^{K-1} U\theta_k \left(\sum_{c=0}^{k}\alpha_{kc}\tilde{\Lambda}^k\right)U^T f = \sum_{k=1}^{K-1}\theta_k\left(\sum_{c=0}^{k}\alpha_{kc}U\tilde{\Lambda}^k U^T\right) f \\
&= \sum_{k=1}^{K-1}\theta_k\left(\sum_{c=0}^{k}\alpha_{kc}\left(U\tilde{\Lambda}U^T\right)^k\right) f = \sum_{k=1}^{K-1}\theta_k T_k\left(U\tilde{\Lambda}U^T\right) f \\
&= \sum_{k=1}^{K-1}\theta_k T_k(\tilde{\mathcal{L}}) f
\end{aligned}
\tag{A.10}
$$

After using Chebyshev polynomial instead of the convolution kernel of the spectral domain, ChebNet does not need the Laplace matrix is to be eigendecomposed. The most time-consuming steps are omitted

A.4 Comparison between Spectral CNN and ChebNet

Assuming that n is the number of nodes.

- The parameter complexity of the SCNN model is very large, and the learning complexity is $O(n)$, which is easy to overfit when there are many nodes. When

dealing with large-scale graph data which usually has more than millions of nodes, it will face great challenges.

- Computing the eigenvalue decomposition of the Laplace matrix is very time-consuming.
- The convolution kernel of ChebNet has only K learnable parameters θ_k, and $K \ll n$, hence their learning complexity is $O(K)$, the complexity of learnable parameters is greatly reduced.
- ChebNet does not need the Laplace matrix to be eigendecomposed, instead it approximate $g_\theta(\mathcal{L})$ with a truncated expansion in term of Chebyshev polynomials $T_k(x)$ of K^{th} order.

A.5 GCN

GCN **?** can be regarded as a further simplification of ChebNet. To reduce the computational complexity, only the first order Chebyshev polynomials are considered, consequently each convolution kernel has only one trainable parameter. Combining with Eq. **??**, we have

$$g_\theta(\Lambda) = \sum_{k=1}^{1} \theta_k T_k(\tilde{\Lambda}) \tag{A.11}$$

Hence,

$$g_\theta(\Lambda) = \begin{bmatrix} \sum_{k=1}^{1} \theta_k T_k(\hat{\lambda}_1) & & & \\ & \sum_{k=1}^{1} \theta_k T_k(\hat{\lambda}_2) & & \\ & & \ddots & \\ & & & \sum_{k=1}^{1} \theta_k T_k(\hat{\lambda}_n) \end{bmatrix} \tag{A.12}$$

In this linear formulation of a GCN we further approximate $\lambda_{max} \approx 2$ Under such approximations, this can simplifies to:

$$\tilde{\mathcal{L}} = \frac{2}{\lambda_{max}} \mathcal{L} - I_n = \mathcal{L} - I_n \tag{A.13}$$

where $\mathcal{L}$ is normalized graph Laplacian $\mathcal{L} = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ Then,

$$f\left(\cdot_G\right) g = \sum_{k=1}^{1} \theta_k T_k(\tilde{\mathcal{L}}) f = \theta_0 T_0(\tilde{\mathcal{L}}) f + \theta_1 T_1(\tilde{\mathcal{L}}) f \tag{A.14}$$

where A is an adjacency matrix of the graph. Accordingly,

$$f\left(\cdot_G\right) g = \left(\theta_0 + \theta_1(\mathcal{L} - I_n)\right) f = \left(\theta_0 - \theta_1(D^{-\frac{1}{2}}AD^{-\frac{1}{2}})\right) f \tag{A.15}$$

Furthermore, to reduce the number of trainable parameters each kernel has only one trainable parameter, we set $\theta_0 = -\theta_1 = \theta$, then we have

$$f\left(\cdot_G\right) g \approx \left(\theta_0 + \theta_1(\mathcal{L} - I_n)\right) f = \left(\theta_0 - \theta_1(D^{-\frac{1}{2}}AD^{-\frac{1}{2}})\right) f = \left(\theta(D^{-\frac{1}{2}}AD^{-\frac{1}{2}} + I_n)\right) f \tag{A.16}$$

where $D^{-\frac{1}{2}}AD^{-\frac{1}{2}} + I_n$ now has eigenvalues in the range $[0, 2]$. Then, only one parameter in convolution kernel can be learned. The number of parameters is greatly reduced, which can reduce the number of parameters to prevent overfitting.

However, repeated application of this operator can therefore lead to numerical instabilities and exploding or vanishing gradients. To alleviate this problem, the following re-normalization trick is introduced.

We add self-loop to A,

$$\tilde{A} = A + I_n \tag{A.17}$$

Correspondingly,

$$\tilde{D}_{i,i} = \sum_{j=1}^{n} \tilde{A}_{i,j} \tag{A.18}$$

Finally,

$$f\left(\cdot_G\right) g = \theta \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} f \tag{A.19}$$

Usually, we write model parameter θ as W, input signal f as X, $\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$, then we have

$$\begin{aligned} f\left(\cdot_G\right) g &= \hat{A} f W \\ &= \hat{A} X W \end{aligned} \tag{A.20}$$

Appendix II

SCRIPTS

B.1 Visualization

```
1  import pandas as pd
2  import numpy as np
3
4  def create_df_correlation(df: pd.DataFrame, method = '
       pearson'):
5      df_pivot_corr = df.corr(method = method)
6      df_corr = pd.melt(df_pivot_corr, ignore_index=False)
       .reset_index().rename(columns = {'index':'node1', '
       variable':'node2'})
7
8      # df_corr = df_corr[df_corr['node1'] != df_corr['
       node2']].copy()
9      df_corr = df_corr.sort_values(['value', 'node1', '
       node2'], ascending=[False, True, True], ignore_index=
       True)
10     df_corr['node1'] = df_corr['node1'].astype(int)
11     df_corr['node2'] = df_corr['node2'].astype(int)
12
13     return df_corr
14
15
16 def filter_df_correlation(df_corr: pd.DataFrame,
       threshold = 0.8):
```

```
17      df_corr_x = df_corr[df_corr['value'].abs() >
    threshold].copy()
18      df_network2 = df_corr.drop(columns = ['value']).
    merge(df_corr_x, on = ['node1', 'node2'], how = 'left
    ')
19      df_network2['value'] = df_network2['value'].where(
    df_network2['value'].isna(), 1).fillna(0)
20      df_network2['value'] = df_network2['value'].where(
    df_network2['node1'] != df_network2['node2'], 0)
21
22      df_pivot_corr2 = pd.pivot_table(df_network2, index=[
    'node1'], columns=['node2'])
23      df_pivot_corr2.columns = [e[1] for e in
    df_pivot_corr2.columns]
24
25      return df_pivot_corr2
```

Listing B.1: Calculate correlation functions

```
1  import matplotlib.pyplot as plt
2  import seaborn as sns
3
4  def plot_corr_heatmap(list_df: list, list_names: list,
       ncols = 2, figsize=(32, 24)):
5      nrows = int(np.ceil(len(list_df)/ncols))
6
7      fig, axs = plt.subplots(nrows=nrows, ncols=ncols,
       figsize=figsize, squeeze=False)
8
9      for i, df in enumerate(list_df):
10         sns.heatmap(
```

```
            df,
            ax = axs[i//ncols][i%ncols],
            vmin=-1, vmax=1, center=0,
            cmap=sns.diverging_palette(20, 220, n=300),
            square=True,
            yticklabels = False,
            xticklabels = False
        )

        axs[i//ncols][i%ncols].set(xlabel="Node", ylabel="Node")
        axs[i//ncols][i%ncols].set_title(list_names[i], fontsize = 36)

    fig.tight_layout()

    return fig
```

Listing B.2: Plot correlation heatmap


```
import matplotlib.pyplot as plt
import seaborn as sns

def corrdot(*args, **kwargs):
    corr_r = args[0].corr(args[1], 'pearson')
    corr_text = f"{corr_r:2.2f}".replace("0.", ".")
    ax = plt.gca()
    ax.set_axis_off()
    marker_size = abs(corr_r) * 10000
    ax.scatter([.5], [.5], marker_size, [corr_r], alpha=0.6, cmap="coolwarm",
```

```
                vmin=0.4, vmax=1, transform=ax.transAxes)
    # font_size = abs(corr_r) * 40 + 5
    font_size = 50
    ax.annotate(corr_text, [.5, .5,],  xycoords="axes
   fraction",
                ha='center', va='center', fontsize=
   font_size)

def plot_corr(df):
  sns.set(style='white', font_scale=1.6)
  g = sns.PairGrid(df, aspect=1.4, diag_sharey=False)
  g.map_lower(sns.regplot, lowess=True, ci=False,
   line_kws={'color': 'black'})
  g.map_diag(sns.distplot, kde_kws={'color': 'black'})
  g.map_upper(corrdot)
```

Listing B.3: Plot pairwise correlarion

B.2 Copula

```
library(VineCopula)
library(data.table)
library(readxl)


readFile<- function(PATH){
  data <- read.csv(PATH)
  # Remove NA
  data <- data[complete.cases(data),]
  data
}
```

```
12
13
14 continue_job_pair <- function(output_file, pair){
15     if (!is.null(output_file)){
16         if (file.exists(output_file)){
17             dfx <- read.csv(output_file)
18             done_at = dim(dfx)[1]
19             final_pair = dim(pair)[2] - done_at
20             output_pair <- pair[, c(done_at+1:final_pair)
    ]
21         } else {
22             output_pair <- pair
23         }
24     }
25     else {
26         output_pair <- pair
27     }
28     output_pair
29 }
30
31
32 uniformiseRandomVariate<- function(data, index_col = "
    Date"){
33   index_data <- data[index_col]
34   # uData <- data.frame(apply(data[,c(-1)], MARGIN=2,
    FUN=pobs))
35   uData <- data[ , !(names(data) %in% c(index_col))]
36   uData <- data.frame(apply(uData, 2, function(c) ecdf(c
    )(c)))
37   uniformData <- cbind(index_data,uData)
```

```
38   uniformData
39 }
40
41
42 preprocess_row_col_lag <- function(data, col1, col2,
       rowlag=NULL, collag=NULL){
43     if (!is.null(rowlag) & !is.null(collag)){
44       print(paste("Error occurs in collag/rowlag
       argument"))
45       break
46
47     }else if (!is.null(rowlag)){
48
49       df_temp <- data[, c(col1, col2)]
50       df_temp[col2] <- shift(df_temp[col2], rowlag)
51       df_temp <- tail(df_temp, -rowlag)
52       df_temp <- na.omit(df_temp)
53
54       u1 <- df_temp[[col1]]
55       u2 <- df_temp[[col2]]
56
57     }else if (is.null(rowlag) & is.null(collag)){
58
59       df_temp <- data[, c(col1, col2)]
60       df_temp <- na.omit(df_temp)
61
62       u1 <- df_temp[[col1]]
63       u2 <- df_temp[[col2]]
64
65     }else if (!is.null(collag)){
```

```
66
67        df_temp <- data[, c(col1, col2)]
68        df_temp[col1] <- shift(df_temp[col1], collag)
69        df_temp <- tail(df_temp, -collag)
70        df_temp <- na.omit(df_temp)
71
72        u1 <- df_temp[[col1]]
73        u2 <- df_temp[[col2]]
74      }
75      list(u1, u2)
76  }
77
78
79  export_to_dataframe <- function(col1, col2, familyname,
      p_indep, AIC, BIC, par, par2, all_numid,
80                                  goftest=FALSE, p_value=
      NULL, stat=NULL, statCvM=NULL,
81                                  statKS=NULL, p_valueCvM=
      NULL, p_valueKS=NULL,
82                                  rowlag=NULL, collag=NULL
      ){
83      if (!is.null(rowlag)){
84          Cop<-data.frame(numid=all_numid, ColVariable=
      col1,RowLagVariable=col2,CopulaBest=familyname,
85                          CopulaBestPvalueIndep=p_indep,
      CopulaBestAIC=AIC,
86                          CopulaBestBIC=BIC,
      CopulaBestParam1=par, CopulaBestParam2=par2)
87      }else if (is.null(rowlag) & is.null(collag)){
```

```
88          Cop<-data.frame(numid=all_numid, ColVariable=
    col1,RowVariable=col2,CopulaBest=familyname,
89                          CopulaBestPvalueIndep=p_indep,
    CopulaBestAIC=AIC,
90                          CopulaBestBIC=BIC,
    CopulaBestParam1=par, CopulaBestParam2=par2)
91      }else if (!is.null(collag)){
92          Cop<-data.frame(numid=all_numid, ColLagVariable=
    col1,RowVariable=col2,CopulaBest=familyname,
93                          CopulaBestPvalueIndep=p_indep,
    CopulaBestAIC=AIC,
94                          CopulaBestBIC=BIC,
    CopulaBestParam1=par, CopulaBestParam2=par2)
95      }
96
97      if(goftest==TRUE){
98          add<-data.frame(CopulaBestPvalueCvM=p_valueCvM,
    CopulaBestPvalue=p_value,
99                          CopulaBestPvalueKS=p_valueKS,
    statCvM=statCvM, statKS=statKS, stat=stat)
100         Cop<-cbind(Cop,add)
101     }
102     Cop
103 }
104
105
106 fitBiCopula <- function(data, pair, goftest=FALSE,
    indeptest=FALSE,
107                         family_list = c
    (1:5,13,14,23,24,33,34),
```

```
108                         rowlag=NULL, collag=NULL,
109                         start_index=NULL, end_index=NULL
    ,
110                         index_col="Date", index_date=
    TRUE,
111                         num_saved = 5, output_file = "
    output.csv", num_task = 0){
112
113   # This function fit bivariate copula
114   # Parameters
115   # ----------
116   # data: Input data
117   # start_index: start index
118   # end_index: end index
119   # goftest: True=run gof test
120   # rowlag: number of day lag by row
121   # collag: number of day lag by column
122
123   Cop <- c()
124   AIC <- c()
125   BIC <- c()
126   emptau <- c()
127   p_indep <- c()
128   family <- c()
129   par <- c()
130   par2 <- c()
131   familyname <- c()
132   tau <- c()
133   beta <- c()
134   col1 <- c()
```

```
135   col2 <- c()
136   all_numid <- c()
137
138   #GoF
139   p_value<-c()
140   stat<-c()
141   p_valueCvM<-c()
142   p_valueKS<-c()
143   statCvM<-c()
144   statKS<-c()
145
146   #indep
147   indep_test <- c()
148
149   if (is.null(start_index)){
150       start_index <- min(input_data[index_col])
151   }
152
153   if (is.null(end_index)){
154       end_index <- max(input_data[index_col])
155   }
156
157   if (index_date==TRUE){
158     start_index<-as.Date(start_index)
159     end_index<-as.Date(end_index)
160     data[index_col] <- as.Date(data[[index_col]])
161     data<-data[(data[[index_col]] >= start_index) & (
      data[[index_col]] <= end_index),]
162   } else {
```

```
163     data<-data[(data[index_col] >= start_index) & (data[
        index_col] <= end_index),]
164   }
165
166   data <- uniformiseRandomVariate(data, index_col)
167   data <- data[ , !(names(data) %in% c(index_col))]
168   num_pair <- dim(pair)[2]
169
170   for (i in c(1:num_pair)){
171     v1<-pair[1,i]
172     v2<-pair[2,i]
173
174     numid <- pair[5,i]
175     all_numid <- c(all_numid, numid)
176
177     u <- preprocess_row_col_lag(data = data, col1 = v1,
        col2 = v2, rowlag=rowlag, collag=collag)
178     u1 <- u[[1]]
179     u2 <- u[[2]]
180
181     set.seed(1234)
182     PwCop<-BiCopSelect(u1, u2, family = family_list,
183                        selectioncrit = "BIC", indeptest=
        indeptest,
184                        rotations = TRUE, method = "mle")
185     col1<-c(col1,v1)
186     col2<-c(col2,v2)
187     AIC<-c(AIC, PwCop$AIC)
188     BIC<-c(BIC, PwCop$BIC)
189     emptau<-c(emptau, PwCop$emptau)
```

```
190     p_indep<-c(p_indep, PwCop$p.value.indeptest)
191     family<-c(family,PwCop$family)
192     par<-c(par,PwCop$par)
193     par2<-c(par2,PwCop$par2)
194     familyname<-c(familyname,PwCop$familyname)
195     tau<-c(tau,PwCop$tau)
196     beta<-c(beta,PwCop$beta)
197
198     if (goftest==TRUE){
199       if (any(PwCop$family==c
    (2,104,114,124,134,204,214,224,234))){
200         set.seed(1234)
201         gof<- BiCopGofTest( u1, u2, family=PwCop$family,
     par=PwCop$par, par2=PwCop$par2, method = "white")
202         p_value<-append(p_value, gof$p.value)
203         stat<-append(stat, gof$statistic)
204         p_valueCvM<-append(p_valueCvM,'NA')
205         p_valueKS<-append(p_valueKS,'NA')
206         statCvM<-append(statCvM,'NA')
207         statKS<-append(statKS,'NA')
208       }
209       else if (any(PwCop$family==c
    (7:10,17:20,27:30,37:40))){
210         set.seed(1234)
211         gof<- BiCopGofTest( u1, u2, family=PwCop$family,
     par=PwCop$par, par2=PwCop$par2, method = "kendall")
212         p_valueCvM<-append(p_valueCvM,gof$p.value.CvM)
213         p_valueKS<-append(p_valueKS,gof$p.value.KS)
214         statCvM<-append(statCvM,gof$statistic.CvM)
215         statKS<-append(statKS,gof$statistic.KS)
```

```
216         p_value<-append(p_value,'NA')
217         stat<-append(stat,'NA')
218       }
219       else{
220         set.seed(1234)
221         gof<- BiCopGofTest( u1, u2, family=PwCop$family,
     par=PwCop$par,method = "kendall")
222         p_valueCvM<-append(p_valueCvM,gof$p.value.CvM)
223         p_valueKS<-append(p_valueKS,gof$p.value.KS)
224         statCvM<-append(statCvM,gof$statistic.CvM)
225         statKS<-append(statKS,gof$statistic.KS)
226         p_value<-append(p_value,'NA')
227         stat<-append(stat,'NA')
228       }
229     }
230
231     if (i%%num_saved == 0){
232
233         msg = sprintf("job%d | %d/%d", num_task, i, dim(
    pair)[2])
234 #         line_task_notify(msg = msg)
235
236         if (goftest==TRUE){
237             Cop <- export_to_dataframe(col1=col1, col2=
    col2, familyname=familyname,
238                                p_indep=p_indep, AIC=AIC, BIC
    =BIC, par=par, par2=par2, all_numid=all_numid,
239                                goftest=goftest, p_value=p_
    value, stat=stat, statCvM=statCvM,
```

```
240                          statKS=statKS, p_valueCvM=p_
   valueCvM, p_valueKS=p_valueKS,
241                          rowlag=rowlag, collag=collag)
242       } else {
243           Cop <- export_to_dataframe(col1=col1, col2=
   col2, familyname=familyname,
244                          p_indep=p_indep, AIC=AIC, BIC
   =BIC, par=par, par2=par2, all_numid=all_numid,
245                          goftest=goftest)
246       }
247
248       write.csv(Cop, output_file, row.names=FALSE)
249
250    }
251
252 }
253
254 if (goftest==TRUE){
255     Cop <- export_to_dataframe(col1=col1, col2=col2,
    familyname=familyname,
256                      p_indep=p_indep, AIC=AIC, BIC=BIC
   , par=par, par2=par2, all_numid=all_numid,
257                      goftest=goftest, p_value=p_value,
    stat=stat, statCvM=statCvM,
258                      statKS=statKS, p_valueCvM=p_
   valueCvM, p_valueKS=p_valueKS,
259                      rowlag=rowlag, collag=collag)
260 } else {
261     Cop <- export_to_dataframe(col1=col1, col2=col2,
    familyname=familyname,
```

```
                        p_indep=p_indep, AIC=AIC, BIC=BIC
    , par=par, par2=par2, all_numid=all_numid,
                        goftest=goftest)
  }
  write.csv(Cop, output_file, row.names=FALSE)
  line_task_notify(msg = sprintf("job %d Done", num_task
    ))

  Cop
}
```

Listing B.4: Fit Copula script

```
install.packages("VineCopula")
library(VineCopula)
```

Listing B.5: petanque.R

```
import rpy2.robjects as ro
import rpy2.robjects as robjects

import numpy as np
import pandas as pd
import itertools

path="/content/petanque.R"
r=ro.r
r.source(path)
```

Listing B.6: Import ‘petanque.R‘ to run in Python

```
def convert2ecdf(x: list) -> np.array:
  x = robjects.FloatVector(x)
```

```
3   fn = r.ecdf(x)
4   output = fn(x)
5   output = np.array(output)
6   return output
```

Listing B.7: Normalized data using ecdf

```
1  import math
2  def rotate_points_origin(x, y, angle):
3    x = np.array(x)
4    y = np.array(y)
5    rad = math.radians(angle)
6    x2 = x*math.cos(rad) - y*math.sin(rad)
7    y2 = x*math.sin(rad) + y*math.cos(rad)
8    return x2, y2
9
10 def rotate_points(x, y, angle, center = (1/2, 1/2)):
11   x0 = center[0]
12   y0 = center[1]
13   x = np.array(x)
14   y = np.array(y)
15   rad = math.radians(angle)
16   x2 = x0 + (x - x0)*math.cos(rad) - (y - y0)*math.sin(
      rad)
17   y2 = y0 + (y - y0)*math.cos(rad) + (x - x0)*math.sin(
      rad)
18   return x2, y2
```

Listing B.8: Rotation points in euclidean domain

```
1  def py_BiCopEst(u1: list, u2: list, family: int):
2      u1 = robjects.FloatVector(u1)
```

```
3     u2 = robjects.FloatVector(u2)
4     cop = r.BiCopEst(u1, u2, family = family)
5
6     d = {key : cop.rx2(key)[0] for key in cop.names if
      key != 'call'}
7     d['taildep'] = d['taildep'][0]
8
9     return cop, d
10
11 def py_BiCopGofTest(u1: list, u2: list, family: int, par
      = 0, par2 = 0, method = 'white'):
12   u1 = robjects.FloatVector(u1)
13   u2 = robjects.FloatVector(u2)
14   gof = r.BiCopGofTest(u1, u2, family = family, par =
      par, par2 = par2, method = method)
15
16   if method == 'white':
17     try:
18       d = {e:float(gof.rx2(e)[0]) for e in ['statistic',
      'p.value']}
19     except:
20       d = {e:float(gof.rx2(e)) for e in ['statistic', 'p.
      value']}
21   elif method == 'kendall':
22     try:
23       d = {e:float(gof.rx2(e)[0]) for e in ['p.value.CvM'
      , 'p.value.KS', 'statistic.CvM', 'statistic.KS']}
24     except:
25       d = {e:float(gof.rx2(e)) for e in ['p.value.CvM', '
      p.value.KS', 'statistic.CvM', 'statistic.KS']}
```

```
26
27    return gof, d
28
29  def py_BiCopSim(N: int, family: int, par: float, par2:
      float = 0):
30    simdata = r.BiCopSim(N, family, par, par2)
31    simdata = np.array(simdata)
32    return simdata
33
34  def py_BiCopSelect(u1: list, u2: list, familyset: list,
      selectioncrit = "AIC",
35                     indeptest = False, level = 0.05,
      weights = None, rotations = True,
36                     se = False, presel = True, method = "
      mle"):
37
38      u1 = robjects.FloatVector(u1)
39      u2 = robjects.FloatVector(u2)
40      familyset = robjects.IntVector(familyset)
41
42      indeptest = robjects.r("TRUE") if indeptest else
      robjects.r("FALSE")
43      weights = robjects.r("NA") if weights is None else
      weights
44      rotations = robjects.r("TRUE") if rotations else
      robjects.r("FALSE")
45      se = robjects.r("TRUE") if se else robjects.r("FALSE
      ")
46      presel = robjects.r("TRUE") if presel else robjects.
      r("FALSE")
```

```
47
48      cop_select = r.BiCopSelect(u1 = u1, u2 = u2,
49                                 familyset = familyset,
50                                 selectioncrit =
    selectioncrit,
51                                 indeptest = indeptest,
52                                 level = level,
53                                 weights = weights,
54                                 rotations = rotations,
55                                 se = se,
56                                 presel = presel,
57                                 method = method
58                               )
59
60      d = {key : cop_select.rx2(key)[0] for key in
    cop_select.names if key != 'call'}
61      d['taildep'] = d['taildep'][0]
62
63      return cop_select, d
```

Listing B.9: Call R funtions in Python

```
1 import plotly.graph_objects as go
2 from plotly.subplots import make_subplots
3 def go_subplot_rotate(x, y, angle_list = [0, 90, 180,
    270], col1 = None, col2 = None, title = None):
4   fig = make_subplots(rows=2, cols=2,
5                       subplot_titles = [f"rotation = {e}
    °" for e in angle_list],
6                       vertical_spacing = 0.15)
7   for i, angle in enumerate(angle_list):
```

```
    x2, y2 = rotate_points(x, y, angle, center = (1/2, 1/2))
    fig.add_trace(go.Scatter(x = x2,
                             y = y2,
                             mode = 'markers',
                             showlegend = False,
                             marker=dict(color='rgba(31, 119, 180, 1)')),
                  row = i//2 + 1, col = i%2 + 1)

  fig.update_xaxes(title = col1, range = [0, 1])
  fig.update_yaxes(title = col2, range = [0, 1])
  fig.update_layout(title = title, width = 500*2, height = 500*2)
  # fig.show()
  return fig

def go_subplot_sim(x, y, family, par, par2, col1 = None, col2 = None, title = None):

  N = len(x)
  simdata = py_BiCopSim(N, family = family, par = par, par2 = par2)

  fig = make_subplots(rows=1, cols=2,
                      subplot_titles = ["<b>Actual</b>", "<b>Simulation</b>"],
                      vertical_spacing = 0.15)

  fig.add_trace(go.Scatter(x = x,
```

```
                            y = y,
                            mode = 'markers',
                            showlegend = False,
                            marker=dict(color='rgba(31,
    119, 180, 1)')),
                row = 1, col = 1)

  fig.add_trace(go.Scatter(x = simdata[:, 0],
                            y = simdata[:, 1],
                            mode = 'markers',
                            showlegend = False,
                            marker=dict(color='rgba(31,
    119, 180, 1)')),
                row = 1, col = 2)

  fig.update_xaxes(title = col1, range = [0, 1])
  fig.update_yaxes(title = col2, range = [0, 1])
  fig.update_layout(title = title, width = 500*2, height
    = 500*1)
  # fig.show()
  return fig
```

Listing B.10: Plot simulation data (from copula) and actual data

```
def go_subplot_sim_set(u1, u2, familyset: list, ncols =
    2, col1 = None, col2 = None, title = None):

    fam_dict = {1: 'Gaussian',
                2: 'Student t',
                3: 'Clayton',
                4: 'Gumbel',
```

```
                5: 'Frank',
                6: 'Joe'}

    N = len(u1)
    d_cop = {}

    cop_select, d_best = py_BiCopSelect(u1 = u1, u2 = u2
, familyset = familyset, rotations = False)
    d_cop["best"] = d_best

    subplot_titles = ["<b>Actual</b>"] + [f"<b>
Simulation ({fam_dict[e]})</b>" if e != d_best['
family'] else f"<b>Simulation ({fam_dict[e]}) **Best
Fitted</b>" for e in familyset] + ["<b>Best Fitted</b
>"]

    nrows = int(np.ceil((len(familyset) + 2)/ncols))

    fig = make_subplots(rows=nrows, cols=ncols,
                        subplot_titles = subplot_titles,
                        vertical_spacing = 0.08,
                    #   horizontal_spacing = 0.1
                        )

    fig.add_trace(go.Scatter(x = u1,
                             y = u2,
                             mode = 'markers',
                             showlegend = False,
                             marker=dict(color='rgba(149,
    63, 181, 1)')),
```

```
31                row = 1, col = 1)
32
33     for idx, family in enumerate(familyset):
34         idx = idx + 1
35         cop, d = py_BiCopEst(u1 = u1, u2 = u2, family =
   family)
36         simdata = py_BiCopSim(N, family = family, par =
   d['par'], par2 = d['par2'])
37
38         d_cop[family] = d
39
40         fig.add_trace(go.Scatter(x = simdata[:, 0],
41                                  y = simdata[:, 1],
42                                  mode = 'markers',
43                                  showlegend = False,
44                                  marker=dict(color='rgba
   (31, 119, 180, 1)')),
45                       row = idx//ncols + 1, col = idx%
   ncols + 1)
46
47     idx = idx + 1
48     sim_best = py_BiCopSim(N, family = d_best['family'],
     par = d_best['par'], par2 = d_best['par2'])
49     fig.add_trace(go.Scatter(x = sim_best[:, 0],
50                              y = sim_best[:, 1],
51                              mode = 'markers',
52                              showlegend = False,
53                              marker=dict(color='rgba(209,
     77, 139, 1)')),
54                 row = idx//ncols + 1, col = idx%ncols +
```

```
    1)
55
56      fig.update_xaxes(title = col1, range = [0, 1])
57      fig.update_yaxes(title = col2, range = [0, 1])
58      fig.update_layout(title = title, width = 500*ncols,
    height = 500*nrows)
59
60      return fig, d_cop
```

Listing B.11: Plot simulation data (from all copulas) and actual data

B.3 ST-CopulaGNN

The scripts uploaded to https://github.com/pitikorn32/ST-CopulaGNN.

Biography

NAME: Pitikorn Khlaisamniang

DATE OF BIRTH: 16 May 1999

PLACE OF BIRTH: Nonthaburi, Thailand

INSTITUTIONS ATTENDED: Chulalongkorn University