DYNAMIC ACTIVITY PREDICTIONS USING GRAPH-BASED NEURAL NETWORKS FOR TIME SERIES FORECASTING

A Thesis
in
Computer Science

Presented to the Faculty of the University
of Missouri–Kansas City in partial fulfillment of
the requirements for the degree

MASTER OF SCIENCE

by
BHUVAN KUMAR CHENNOJU

Kansas City, MO, USA

Kansas City, Missouri
2023

DYNAMIC ACTIVITY PREDICTIONS USING GRAPH-BASED NEURAL NETWORKS FOR TIME SERIES FORECASTING

Bhuvan Kumar Chennoju, Candidate for the Master of Science Degree

University of Missouri–Kansas City, 2023

ABSTRACT

Time series forecasting is a vital task in numerous fields and traditional methods, machine learning models, and neural graph networks have been employed to improve prediction accuracy. However, these techniques need to be revised in understanding inter-dependencies and establishing long-term dependencies when dealing with a network of time series, such as predicting energy demand on interconnected grids. To tackle these challenges, this thesis introduces a framework implementing Attention-based Temporal Graph Convolutional Networks (ATGCNs) that enables holistic treatment of a group of time series while learning inter-dependencies and facilitating message passing for enhanced model efficiency. The major contribution of this thesis lies in developing graph embedding algorithms that convert Microbusiness density data into graph data, considering the spatial distance and time series for the proposed ATGCNs model, enabling dynamic activity predictions.

The proposed framework is evaluated through experiments using a U.S. Microbusiness density dataset from the GoDaddy Open Survey. The results reveal that ATGCNs outperform traditional time series statistics and machine learning methods in various evaluation metrics, demonstrating comparable forecasting performance to conventional time series forecasting while addressing network scalability and dynamic nature. Additionally, real-time prediction visualizations based on Tableau were developed to showcase the dynamic nature of predictions in the U.S. Microbusiness density domain.

In conclusion, this study's findings highlight the potential advantages of employing graph-based neural networks for time series forecasting, suggesting that incorporating additional data sources could improve prediction accuracy. As future work, transfer learning with ATGCNs will be applied to new domains such as climate prediction or energy demand on interconnected grids. Furthermore, the graph-embedding algorithm and visualization techniques developed in this project will be applied to new domains and datasets across different domains.

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the School of Science and Engineering, have examined a thesis titled ”Dynamic Activity Predictions using Graph-based Neural Networks for Time Series Forecasting” presented by Bhuvan Kumar Chennoju, a candidate for the Master of Science degree, and hereby certify that in their opinion it is worthy of acceptance.

Supervisory Committee

Yugyung Lee, Ph.D., Committee Chair
Department of Computer Science Electrical Engineering, UMKC

Reza Derakhshani, Ph.D.
Department of Computer Science Electrical Engineering, UMKC

Sejun Song, Ph.D.
Department of Computer Science Electrical Engineering, UMKC

CONTENTS

ILLUSTRATIONS

TABLES

ACKNOWLEDGEMENTS

This fruitful journey as a researcher would not have been possible without my professors, mentors, family, and friend's support. I am deeply indebted to my academic advisor, and supervisor, Dr. Yugyung Lee, for her continuous guidance, support, and valuable insights throughout my research journey. Her expertise, patience, and encouragement have been invaluable to me, and I am immensely grateful for her mentorship.

My deepest gratitude to my professor, and mentor, Dr. Reza Derakhshani, for his wisdom and guidance in an influential way in shaping my mindset as a student, teacher, and researcher. I also extend my heartfelt thanks to the School of Computing and Engineering faculty members and staff at the University of Missouri-Kansas City for providing a stimulating and supportive environment that enabled me to pursue my research goals. I also thank committee member Dr. Sejun Song for his constructive advice and feedback.

My family, parents, and brother! Their unconditional love and support made this journey possible. They have always been my biggest cheerleaders, motivating me to pursue my dreams and helping me through the ups and downs of life, and I am forever indebted to them. The support and encouragement of my dear friends have also been invaluable to me.

Finally, I express my sincere appreciation to all those who have contributed to my research project in one way or another. Your support, encouragement, and constructive feedback have been instrumental in shaping my research and helped me to achieve my goals. I am grateful to you all.

CHAPTER 1

INTRODUCTION

Over the past two decades, with the widespread of digital technologies such as smartphones, social media platforms, and IoT devices, exponential growth in data generation has been witnessed in several industries [2–5]. With rapid digitalization and industry 4.0, the time-dependent data collection to make data-driven decisions has gained prominence [4]. In recent years, many research articles have been published on time series because of their applications [6, 7], such as stock market predictions, web, transportation traffic predictions, climate studies, wind pattern predictions, etc. Several techniques have been employed in the field of time series forecasting, including Winters exponential smoothing [8], the Autoregressive Integrated Moving Average (ARIMA) model [9], multiple regression, and artificial neural networks (ANNs) [10]. These approaches have gained significant popularity due to their ability to capture the trend and seasonal variations often present in time series data [11]. But, past values, external factors outside the survey, outliers, or hidden connections can impact the future of forecasting metrics. Factors such as hidden dependencies on external influences, outliers, and significant events can influence forecasting accuracy. So, despite the extensive research on individual time series forecasting with statistical methods and classical machine learning, there is a need for more understanding of the relationship between different time series of a similar domain. Given the critical role played by time series data and underlying interdependencies in numerous disciplines, the present dissertation is devoted to graph-based time series modeling and forecasting, with extensive cross-comparison with statistical modeling, machine learning modeling, and deep learning modeling-based time series forecasting.

1.1 Background

The recent generation of spatiotemporal data has revolutionized how we study and understand the world. With the proliferation of smartphones, Internet of Things (IoT) devices, and sensors, vast amounts of spatiotemporal data are being generated at an unprecedented rate. This data includes information about the location, movement, and behavior of people, vehicles, objects, and environmental conditions such as temperature, humidity, and air quality [12].

The importance of studying spatiotemporal data concerning time series forecasting must be considered. Time series forecasting is critical in many fields, including finance, transportation, energy, and environmental science. Accurate forecasting of spatiotemporal data can provide valuable insights into the behavior and trends of complex systems, leading to better decision-making and resource allocation.

Two major categories of data-driven methods for time series analysis are classical statistical models and machine learning models. Autoregressive integrated moving average (ARIMA) [13] and its variants are classic statistical models commonly used for time series analysis [14]. However, these models are limited by their assumption that time sequences are stationary and do not consider spatiotemporal correlation. As a result, they have a limited ability to represent highly nonlinear time series data. In recent years, machine learning methods such as the k-nearest neighbors algorithm (KNN), support vector machine (SVM), and neural networks (NN) have gained popularity for such data due to their higher prediction accuracy and ability to model complex data.

Although these algorithms gained popularity for there non-linear generalizations, According to the research conducted by Zhang and Qi, [15], it was found that Neural Networks working alone are not effective in modeling the trend and seasonality accurately. This suggests that the performance of forecasting with Neural Networks can be enhanced by detrending or deseasonalizing the raw time series before modeling it. In other words, it is necessary to preprocess the data to remove the trend and seasonality components before feeding it into the neural network to obtain better forecasting results. This highlights the importance of accurate data preprocessing and feature engineering in improving the accuracy of time series forecasting with Neural Networks.

To improve forecasting accuracy, researchers have introduced several strategies to overcome the limitations of traditional RNN-based networks, including their difficulty in training and computational heaviness. To address these issues, we adopted a novel deep learning architecture called spatiotemporal graph convolutional networks [12]. This architecture uses a general graph to model the general spatial-temporal network and employs a fully convolutional structure on the time axis. It comprises several spatiotemporal convolutional blocks that combine graph convolutional layers and convolutional sequence learning layers to model both spatial and temporal dependencies.

1.2 Problem Statement

The objective of this master's thesis is to develop a framework that will enhance our understanding of interdependent time series forecasting. The framework will encompass conventional naive methods, statistical methods, machine learning methods, and graph-based deep learning to better understand time series forecasting as a whole cluster rather than treating them as mutually exclusive independent time series.

To illustrate this concept, let's consider the prediction of the energy demand of each state in the United States or the stock market price of a company about its subsidiaries. By treating each time series as an individual entry, the prediction will be biased toward that entry. However, by treating this network of entries as a group with interdependency relationships, the prediction accuracy can be improved by weighted messaging within the group. Therefore, its worth looking in this direction to explore this hypothesis, validate the results, and conclude the applicability of this framework to a general time series forecasting framework.

1.3 Objectives

This thesis addresses the research question by considering the relevant aspects discussed in the preceding section. To fully answer the research question, the following objectives will be pursued:

- Build and analyze the future observations as individual entries.
- Create baseline models for comparison purposes.

- Develop a framework to cross-compare the traditional methods of time series forecasting, machine learning methods, and temporal graph networks.
- Convert the problem statement in hand to graph network.
- Create the benchmark for the spatial-temporal framework to check the validity of the current problem.
- Analyze and evaluate the proposed framework's performance compared to the baseline model(s), assessing the effectiveness of the model and the performance of the temporal graph networks.
- summarizes the observations with validation of the thesis hypothesis.

1.4 Limitations

Few limitations that are observed while researching as follows:

- The data set in considered is significantly reasonable in terms of spatial dimension, whereas having only two and has cycles of seasonal data. This means we cannot answer if the prediction perform ace could be improved with more data in proportion to the spatial dimension.
- The critical factors for time series forecasting problems, like the holiday effect, will not be our score in this thesis, as the data is once in month collected. Although Facebook's prophet uses this idea of the holiday effect, it can only be effective as the data have only monthly observations.

- The speed and model optimization is not in the scope of the work, and the core spatiotemporal network architecture is adopted from preexisting work [12]. So we will not validate the efficiency of networks in the thesis but focus on the application of the idea.

1.5 Thesis Organization

This thesis is divided into five chapters. The current introduction chapter is the first chapter. The second chapter provides theoretical context, focusing on related work in time series data and time series forecasting techniques, including naive methods, naive statistical methods, machine learning methods, and Graph-based deep learning. The third chapter introduces the suggested framework, which includes three distinct phases and a summary of contributions. The fourth chapter examines the outcomes and judgments of our work in a broader context. Finally, the fifth chapter finishes the thesis by summarizing the findings in conclusion and making recommendations for further research.

CHAPTER 2

RELATED WORK

The present chapter serves as an introduction to the background of our research. Our research pertains to Time Series Forecasting. Therefore, its relevant theory was discussed at a foundational level. The structure of this chapter is as follows: 1) We will provide an overview of the time series forecasting problem and some recent advances in this area. 2) We will discuss Artificial Neural Networks (ANNs), specifically Long Short-Term Memory (LSTM) networks, which are utilized in our study. 3) We will introduce Graph Convolutional Neural Networks, specifically Temporal graph convolutional networks, which are crucial for the study.

2.1 Time Series Data:

Time series data is a type of data that represents a sequence of observations collected at regular intervals over time. It is a set of data points indexed in time order, usually with equal gaps between the time points.

Mathematically speaking, a time series is a sequence of Y comprising an ordered sequence of data points $Y = y_1, y_2, ... y_t$, where each y_i represents a data point at a specific time i. Figure 1shows three different time series from our US counties data for periods from 2019 to 2022 w.r.t temporal variable Y.

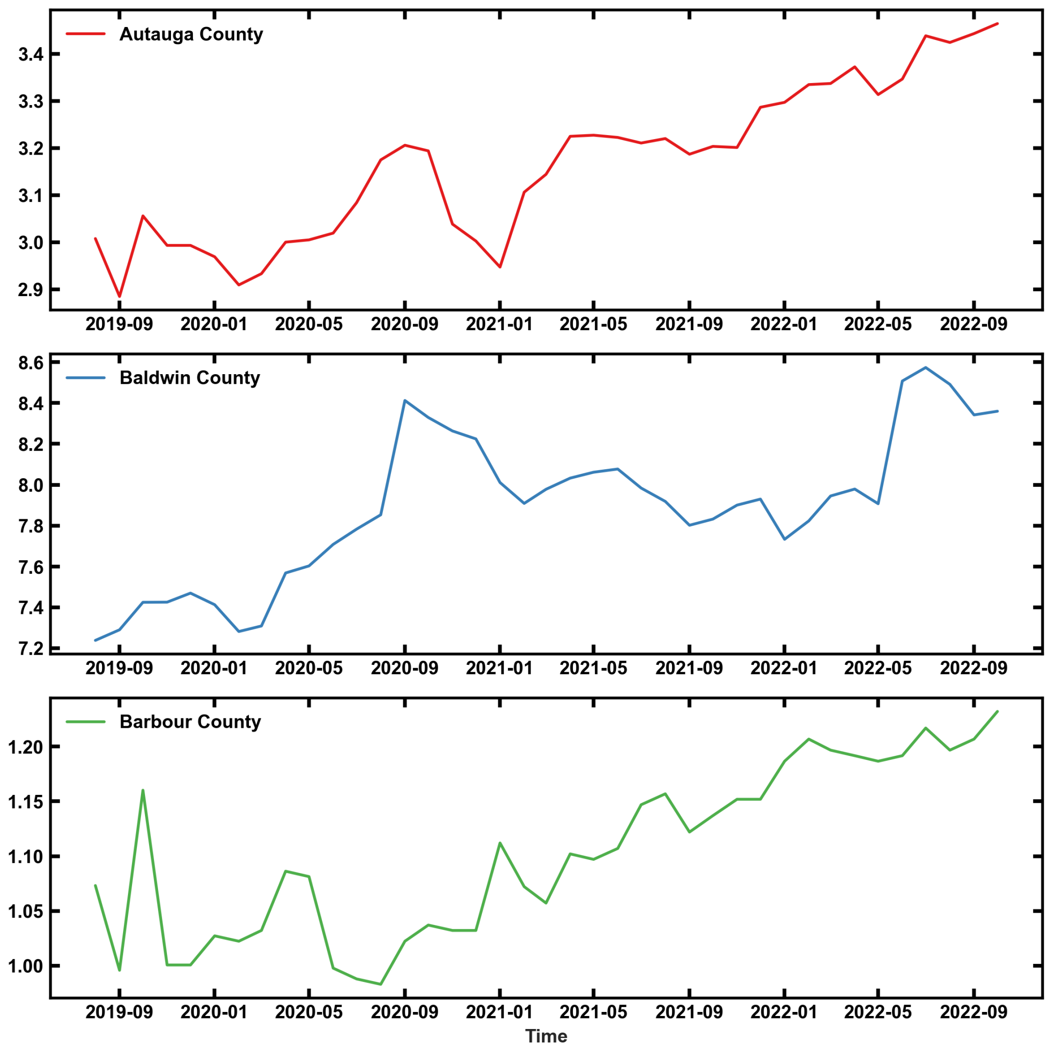


Figure 1: Time series from the Microbussinesses density data.

Every time series data can be decomposed into four primary components: trend, cycle, seasonal variations, and outliers. Figure 2 shows these components. The trend component represents the time series general upward or downward tendency over time. The cycle component captures medium-term changes in the series, often driven by specific circumstances. The seasonal variations represent regular high and low values patterns within a year, including seasonal, quarterly, monthly, or daily cycles. The outlier component identifies data points that deviate significantly from the overall patterns and are often attributed to unpredictable factors. Details of trends and seasonality below.

Table 1: Frequency of Different Series Types

Series Types	Frequency
Yearly	1
Quarterly	4
Monthly	12
Weekly	52

- **Trend:** Trend in time series refers to the long-term pattern or direction of the data. It reflects the underlying behavior of the series and can help identify changes or shifts in the overall pattern. Trends can be positive, negative, or flat, depending on whether the data increases, decreases, or stays relatively constant. Figure 2 shows the increasing trend for the given time series.
- **Seasonality:** Seasonality is a characteristic of time series data that refers to a pattern of behavior that repeats itself regularly, usually within one year or less. Seasonality

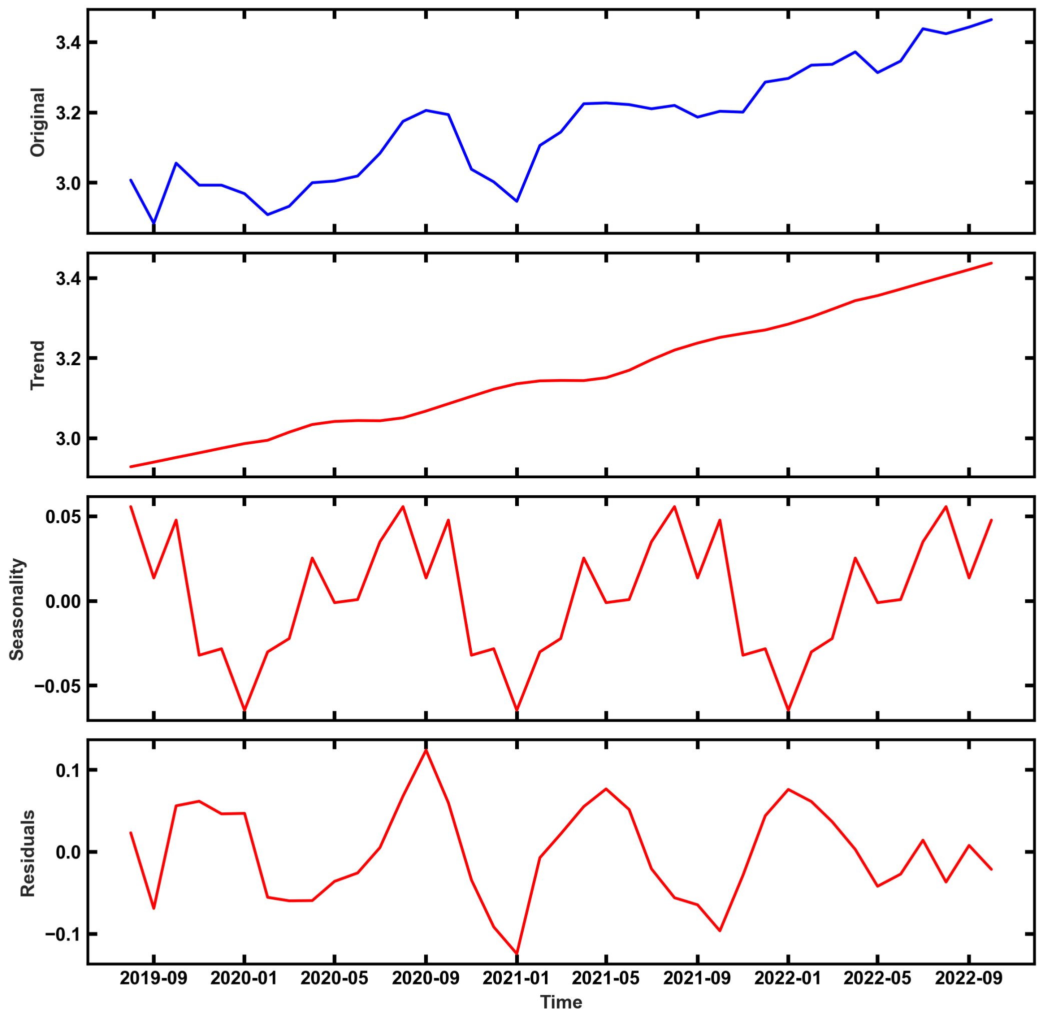


Figure 2: Trend, Seasonality, and Residuals Decomposition of one of the time series, from the US Microbusiness density data.

can be seen in several things, such as stocks, weather patterns, fashion, etc. For Instance, sales of winter coats tend to increase in the months leading up to winter and decrease in the warmer months, while ice cream sales tend to grow in the summer and fall in the colder months. In common practice, the seasonality was addressed with the frequency of the time series, and well know the frequency of the time series is [12] given in Table 1. Figure 2 shows the seasonality of frequency 12, which is of monthly type seasonality.

- **Residuals:** Residuals are differences between original data, the sum of trend, and seasonality components. Residuals are essential to time series modeling as they provide information about the critical things missed by the movement, and seasonality, such as holidays and events. Figure 2 shows the residuals of the time series.

2.2 Time Series Forecasting

Predicting future observations based on historical data is highly valued in numerous applications, as it allows for proactive decision-making to mitigate risks and minimize potential losses. Examples of time series data include sales figures, stock price predictions, weather pattern predictions, earthquake occurrences, and foot traffic data. The main goal of effective time series forecasting is to forecast future observations as accurately as possible. Many methods and techniques exist to forecast a time series: time series decomposition, statistical forecasting, exponential smoothing techniques, machine learning, artificial neural networks, and temporal convolutional graph networks.

2.2.1 Time Series Decomposition

Time series data often exhibit complex patterns that can influence their forecasting accuracy. Therefore, it is essential to decompose a time series into its underlying components to understand these patterns better and improve the accuracy of the forecasts. By breaking down the time series into several parts that represent distinct pattern categories, such as trend, seasonality, and residual variation, it is possible to analyze and model each component separately, which can lead to better insights into the behavior of the data. Composing a time series can also help identify anomalies or outliers, impacting the forecasting models' overall accuracy. Several methods exist for time series decomposition, including classical and seasonal decomposition of time series (STL).

- **Classical Decomposition** In classical decomposition, the time series aims to decompose into four components: trend, seasonality, cycle, and residuals. Trend, seasonality, and residuals are explained in Section 2.1. Two fundamental ways to split a time series are additive and multiplicative [16].

  Mathematically, the classical additive decomposition can be represented as:

  $$Y(t) = T(t) + S(t) + C(t) + e(t)(2.1) \tag{2.1}$$

  Alternatively, the classical multiplicative decomposition can be represented as:

  $$Y(t) = T(t) * S(t) * C(t) * e(t)(2.2) \tag{2.2}$$

- **Seasonal Decomposition** In seasonal decomposition, time series data is into its underlying components, including the seasonal, trend, and irregular components [14].

Unlike classical decomposition, seasonal decomposition is specifically designed to handle time series data with strong seasonal patterns. The seasonal component represents the regular fluctuations within the data, while the trend component represents any long-term upward or downward movements. The irregular component represents any random fluctuations or noise that the seasonal or trend components cannot explain.

2.2.2 Statistical Forecasting Methods

Statistical forecasting methods are based on historical data and mathematical models to identify patterns and trends in time series data and predict future values [16]. These methods have been widely used in various industries, including finance, manufacturing, and transportation, to make informed decisions about future planning and resource allocation. Generally, as a rule of thumb, these statistical forecasting methods are used as the benchmark [17–20] for time series forecasting. In most time series forecasting, these statistical methods effectively make benchmarks and cross-comparisons [20].

- **Naive Method** In the naive method, the past observation is assumed as the future forecasting. Mathematically speaking, for a time series Y with observations y_i at $i = 1, 2, 3, ...t$, then predict for $t + h$ is defined as:

$$\hat{y}(t + h|t) = y(t) \tag{2.3}$$

where $\hat{y}(t + h|t)$ represents the time series forecast at time $t + h$, based on the information available at time t, $y(t)$ is the time series value at time t,m is the seasonal period, and h is the forecast horizon. With this idea, this method acts as a suitable base as ground zero to start the predictions; like null value accuracy in the classification, this method gives the initial frame of reference to improve the model's effectiveness.

- **Seasonal Naive Method** The naive seasonal method uses the observation from the same season in the previous year as the forecast for the upcoming season without considering any trend or seasonal components [21]. This approach can be expressed mathematically as follows:

$$\hat{y}(t+h|t) = y(t+h-m(k+1)) \quad where \quad k = \left\lfloor \frac{h-1}{m} \right\rfloor \tag{2.4}$$

  where $\hat{y}(t+h|t)$ represents the forecast of the time series at time t+h, based on the information available at time t, $y(t)$ is the value of the time series at time t, m is the seasonal period, h is the forecast horizon, and k is an integer that determines how many seasonal periods need to be subtracted from $t+h$ to find the corresponding observation from the previous year.

- **Autoregressive Integrated Moving Average** Autoregressive integrated moving average (ARIMA) is a famous [22–24] generalization of a simpler Autoregressive Moving Average (ARMA) model, which can handle the non-stationary time series data by including the integration component. ARIMA model is primarily designed to handle stationary time series data [21]. However, non-stationary data can be transformed into stationary data by taking the first, or second-order differences.

This process is called differencing, and itâs a popular technique to convert non-stationary to stationary data. ARIMA model can be defined by three parameters, namely p,d, and q. The parameter p represents the order of the autoregressive (AR) component, d represents the degree of difference needed to make the time series stationary, and q represents the order of the moving average (MA) component.

- **Prophet** Facebook Prophet is a forecasting system developed by Facebook's data science team based on a decomposable time series model. The model has three main components: growth, seasonality, and holidays. The growth function captures the trend in the time series, seasonality models periodic changes due to weekly or yearly seasonality, and holidays and special events are considered as another component. These three components are combined using a generalized additive model (GAM) framework. Prophet relies on linear/non-linear growth, Fourier series, and dummy variables to model the growth, seasonality, and holiday components. The system is mainly used for daily frequency time series with at least one year of historical data, but it can also be applied to time series with other frequencies. The Prophet package is implemented in R and Python. Although Prophet is easy to use, it has some limitations. For example, it is not designed for multivariate time series data and does not consider features other than seasonality or special events.

2.2.3 Exponential Smoothing Methods

Exponential smoothing, first introduced by Brown in 1959 [25] and later developed by Holt [26] and Winters [27], has been widely recognized as one of the most successful and practical forecasting methods. This method involves computing weighted averages of historical observations to generate future forecasts. The weights assigned to the observations decay exponentially over time, giving more weight to recent observations and less weight to older observations. This feature allows the method to adapt to changes in the time series pattern and efficiently generate reliable forecasts for a broad range of time series. Consequently, various industries commonly adopt exponential smoothing[19].

The Holt-Winters method for forecasting time series data with seasonality can be expressed mathematically as follows:

$$s_0 = y_0 \tag{2.5}$$

$$\hat{y}_t = \alpha(y_t - st - m) + (1 - \alpha)(l_{t-1} + b_{t-1}) \tag{2.6}$$

$$b_t = \beta(\hat{y}_t - \hat{y}_{t-1}) + (1 - \beta)b_{t-1} \tag{2.7}$$

$$s_t = \gamma(y_t - \hat{y}_t) + (1 - \gamma)st - m \tag{2.8}$$

For the additive method:

$$\hat{y}_{t+1} = l_t + b_t + st + 1 \tag{2.9}$$

For the multiplicative method:

$$\hat{y}_{t+1} = l_t * b_t * st + 1 \tag{2.10}$$

Where s_0 is the initial seasonality estimate, $\hat{y}_t$ is the level estimate at time t, b_t is the trend estimate at time t, s_t is the seasonality estimate at time t, α, β, and γ are the smoothing parameters for level, trend, and seasonality, respectively, m is the frequency of the seasonality, y_t is the actual observation at time t, y_{t+1} is the forecast for the next time period.

2.2.4 Machine Learning Methods

Machine learning methods have gained significant attention [20] in time series forecasting due to their capability to model complex patterns in data. These methods include traditional techniques such as regression, decision trees, and neural networks, as well as more advanced procedures like support vector machines, random forests, and deep learning [22, 28–31] One of the significant advantages of machine learning methods is their ability to handle high-dimensional and non-linear data, a common characteristic of many time series datasets. Additionally, machine learning algorithms can automatically identify relevant features and learn from historical data to improve forecasting accuracy. Different regression algorithms, such as Linear Regression, Support Vector Regression, and decision trees, are applied in the various application of time series forecasting [22, 28–31].

- **Linear Regression** Linear regression is a method used to identify a linear relationship between the dependent and independent variables. The dependent variable is the variable we want to predict, while the independent variable(s) are used to make predictions. Simple and multivariable regression are the two main types of linear regression, with the former useful for finding the relationship between one independent variable and the dependent variable and the latter involving multiple independent variables. To optimize a cost function, linear regression uses gradient descent. Linear regression is also used in time series forecasting, as seen in previous studies [32].

Mathematically speaking, if X is a dependent variable, Y is an independent variable; the model assumes that the relationship between X and Y is linear, i.e., Y is a linear function of X plus an error term ϵ.

$$\overline{Y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \epsilon \tag{2.11}$$

Where $\beta_0, \beta_1, \beta_2, \ldots, ...\beta_n$ are the regression coefficients or parameters to be estimated, and ϵ is the error term, representing the random variability in the data that the model does not explain. The goal of linear regression is to estimate the values of $\beta_0, \beta_1, \beta_2, \ldots, ...\beta_n$ that best fit the data by minimizing the sum of the squared errors between the predicted values of Y and the actual values of Y. This is often done using the method of least squares. This thesis used the root mean square error as a metric for this model.

- **Support Vector Regression** Support Vector Regression (SVR) is a machine learning algorithm like Support Vector Machines (SVM) used to predict continuous values. The algorithm constructs a hyperplane to maximize the margin and minimize the error. The hyperplane is then used to create a decision boundary so the support vectors are within the boundary lines. SVR can be formulated as a constrained optimization problem, where the objective is to minimize the prediction error subject to a tolerance margin. The solution is obtained by finding the Lagrange multipliers that satisfy the Karush-Kuhn-Tucker conditions. The support vectors are those data points that are closest to the boundary. The best-fit line is the hyperplane with the maximum number of points in the decision boundary. SVR has been widely used in time series prediction and analysis, such as financial and stock time series forecasting [33, 34].

  The mathematical Formulation of SVR is as follows:

  Given a set of training data $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ where x is the input feature vector, and y is the corresponding target value, SVR seeks to find a function $f(x)$ that predicts the target value y for a new input x. The function $f(x)$ is defined as:

  $$f(x) = w \cdot x + b \tag{2.12}$$

  where w is a weight vector, and b is a bias term. SVR aims to find the optimal values of w and b such that the distance between the predicted value $f(x)$ and the true target value y is minimized, subject to a tolerance margin ϵ.

This can be expressed as the following optimization problem:

$$\frac{1}{2}|w|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{2.13}$$

$$y_i - f(x_i) \leq \varepsilon + \xi_i \tag{2.14}$$

$$y_i - f(x_i) \leq \epsilon + \xi_i f(x_i) - y_i \leq \epsilon + \xi_i^*, where\, \xi_i, \xi_i^* \geq 0 \tag{2.15}$$

Where ξ_i and ξ_i^* are slack variables, and C is a hyperparameter that controls the trade-off between achieving a smaller margin and allowing more violations to the margin. The optimization problem can be solved using various methods, such as quadratic programming or gradient descent.

- **Random Forest Regression**

Random Forest Regression is a popular ensemble learning algorithm for classification and regression tasks. The algorithm uses multiple decision trees to improve predictive accuracy, where each tree is trained on a random subset of features and samples selected with replacement using bagging. This approach prevents overfitting and provides a reliable and accurate prediction. The Random Forest model is computationally efficient, simple to use, and requires only a few parameters to tune. It is a powerful tool widely used in applied machine learning and time series prediction [35, 36].

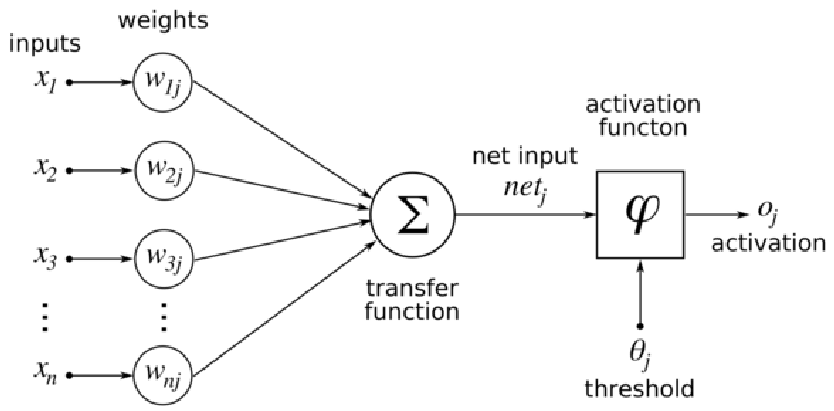


Figure 3: Schematic diagram of a perceptron, a simple neural network architecture with one input layer, one output layer, and no hidden layers. The input layer receives inputs from external sources or other nodes in the network, and the output layer produces the final output of the perceptron. The weights and bias of the perceptron are adjusted during training to improve the accuracy of its predictions. [1]

2.2.5 Artificial Neural Networks

Artificial Neural Networks, or ANNs for short, are sophisticated computational tools designed to extract patterns, unknown structures, and insights from data. They are modeled on the structure and function of biological neural networks, employing interconnected processing units known as artificial neurons to process input signals and produce a weighted sum of outputs. These outputs are then passed through a non-linear activation function to generate the result. The Perceptron is a classic example of an artificial neuron, and it is widely used as a supervised learning algorithm for binary classification problems. Figure **??** shows the popular Rosenblatt perceptron.

The ability of ANNs to analyze complex data patterns attracted the scientific community to its applicability the time series forecasting. Predominantly, time series forecasting is dominated by Recurrent neural networks, Gated recurrent neural networks, and long short-term memory (LSTM). RNNs [37] and LSTMs [38] have become popular with time series forecasting. RNN is a Neural network that takes the input of sequence and output sequence based on the lags in the future. Similarly, LSTM is a special RNN with skip connections and works on memory-based output.

The most basic ANN type of ANNs is a feedforward neural network (FFNN) or a multi-layer perceptron (MLP). It consists of multiple perceptions, which aim to approximate a function f from the known data x and the target data y. This mapping can be defined as $y = f(x, w)$, where the main objective of learning is to obtain the optimal weights w for the best function approximation. In a neural network, the layers represent the mapping functions, and the number of stacked layers determines the depth of the model. The term ”Deep Learning” refers to models with many hidden layers.

When training a machine learning model, the goal is to ensure that the model's output, represented as $f(x)$, is as close as possible to the real target values. This process involves utilizing both the training data and test data. To achieve this, we define a set of input-output pairs $(x; y)$, a blueprint for the model to generate the corresponding y value based on the given x value. Optimizing the model's parameters during training can reduce the difference between the predicted output and target values, allowing the model to generalize accurately to new, unseen data.

To ensure that a machine learning model is accurate in its predictions, a mechanism is required to help the model determine the quality of its calculations. This mechanism measures the difference between the calculated output $\hat{y}$ and the actual output y. In other words, we need to use a loss function to quantify this difference during the training process and provide feedback to the model on its performance. The loss function is a mathematical function that takes the difference between predicted and actual outputs as input and outputs a single value representing the model's error. The goal of training the model is to minimize the loss function, which means minimizing the difference between the predicted and actual outputs. The loss function is usually optimized with a gradient-based algorithm; Stochastic Gradient Descent is a commonly used optimization algorithm.

- **Recurrent Neural Network** RNNs (Recurrent Neural Networks) are a type of neural network that address the issue of dependency between input sequences. Unlike traditional neural networks, which assume all inputs are independent, RNNs utilize sequential information to compute the output. This is particularly useful for applications such as language modeling, where the sequence of words is important for predicting the next word in a sentence. RNNs use a hidden state updated at each time step, allowing the network to maintain a memory of previous inputs. The output at each time step is generated based on the current input and the previous hidden state.
- **Long Short-Term Memory Network** LSTM stands for Long Short-Term Memory, a Recurrent Neural Network (RNN) architecture designed to solve the vanishing

gradient problem in standard RNNs. In an LSTM network, each neuron, also called a memory cell, has three gates: the input gate, the forget gate, and the output gate. The input gate regulates the flow of new information into the memory cell, the forget gate controls the retention or deletion of old information, and the output gate decides how much information to output from the memory cell to the next neuron in the sequence. By controlling the flow of information, the LSTM can selectively remember or forget information from the past, making it effective in modeling sequential data with long-term dependencies.

2.2.6 Temporal Graph Neural Networks

Temporal Graph Neural Networks (TGNs) [39–42] are architecture that operates on time-evolving graph data. These networks extend the traditional Graph Neural Networks (GNNs) by incorporating the temporal dimension, allowing them to model how graph structures and node attributes change over time.

TGNs operate on a sequence of graphs, where each graph represents the system's state at a particular point in time. In this sequence, each node has a set of attributes that can change over time, and edges can also appear or disappear over time. TGNs aim to learn a function that can predict future states of the graph, given the current state and historical information. TGNs use a combination of convolutional graph layers and temporal convolutional layers to process the input data. Graph convolutional layers operate on each graph individually, allowing the model to learn how node features and graph structure affect each other. Temporal convolutional layers work on the sequence of graphs, allowing the model to learn how the graph evolves. One popular variant of TGNs is the Temporal Graph Convolutional Network (TGCN), which uses a shared graph convolutional layer for each time step and a temporal convolutional layer to capture temporal dependencies. Another variant is the Evolving Graph Convolutional Network (E-GCN). It adds an attention mechanism to weigh the contributions of different time steps based on their relevance to the current prediction task. TGNs have shown promising results in various applications, including predicting social interactions, traffic flow, and financial markets. However, they can be computationally expensive due to the need to process a sequence of graphs, and there is ongoing research to improve their scalability and efficiency.

CHAPTER 3

METHODS AND METHODOLOGIES

In this chapter, we will formulate the methodologies to validate the thesis hypothesis and ways to answer the research question from Chapter 1.3. To achieve this goal, we will first review the literature on the current state-of-the-art in time series forecasting and temporal graph networks in chapter 2. This literature review will help us identify the strengths and weaknesses of existing models and highlight the gaps in the current research.

Now, we'll be able to propose a framework for a model that combines the strengths of conventional neural networks and temporal graphs. The proposed model will address the challenges in time series forecasting, such as handling long-term dependencies and predicting uncertain events. To work with such model, the main hurdle is the data, and graph conversions, to explore the effectiveness of the model.

So In this chapter, detailed information about the data, data analysis, structured data processing to graph data, model, evaluation metrics, and training paradigm. Later, we will evaluate the proposed model's performance by conducting experiments on real-world time series datasets. The experiments will compare the performance of the proposed model against the conventional neural network and other state-of-the-art time series forecasting models.

Finally, we will analyze the experiments' results to conclude the proposed model's effectiveness and the potential benefits of temporal graphs for time series forecasting. This analysis will provide insights into the practical applications of the proposed model and its potential to improve the accuracy and reliability of time series forecasting.

3.1 Data Collection

3.1.1 U.S. Microbussiness Density Data

In this research, the data used is the microbusiness density of the united states, where Microbusiness density is the number of microbusinesses per 100 adults in a geographic area. This data was collected by Go Daddy surveys for this study in 2018 to analyze the economic impact of micro-businesses in the United States. Go Daddy worked with economists and data scientists from various institutions to analyze data from over 30,000 zip codes, 3,000 counties, and 900 city regions. The microbusiness density was measured by counting the active Go Daddy-registered domains in a geographic area, including commercial ventures, nonprofits, cause-oriented, and other sites. The data was further divided to identify highly active microbusinesses in each region. The study combined data from various sources, including the U.S. Census Bureau, Bureau of Labor Statistics, and Economic Innovation Group, to model the effects of online microbusinesses on different economic outcomes, such as regional prosperity scores, median household income, and unemployment rates. The regression models controlled for additional variables like broadband access, education levels, age, demographics, population, and occupational data. The dataset provides aggregated results at the Core-Based Statistical

Area (CBSA), County, and State levels, including monthly microbusiness density measurements. The data needs to catch up by one calendar quarter due to business reasons, and the current dataset began in August 2019 [43].

3.2 Data Analysis

It is essential to conduct a thorough analysis to ensure that the data collected is suitable for the intended research purposes. This analysis will help determine whether the data is complete and accurate and identify potential outliers or anomalies. Once the data has been verified, appropriate analysis metrics must be chosen to evaluate the data generated from the experiment effectively. The selected metrics should be able to provide meaningful insights into the data and help answer the research questions or hypotheses. This section of the research process is crucial in ensuring the data analysis is accurate and reliable, enhancing the credibility and validity of the research findings.

3.2.1 Data Analysis of U.S. Microbussiness Density Data

This data focuses on microbusiness density values at various levels from state to county and the data available from 08/2019 to 10/2022. So 39 monthly data are collected for all the states to the county level. There are 52 states and 3135 counties with 39 months of time series data in the collected data. So in terms of individual entities, there are 3135 unique time series overall. Overall, there is all the information. For each state, census data was collected from the open-census data hub. This data was later used together to give additional information about each state. The distribution of the microbusiness density values is shown in Figure 4, where the mean value falls around 2.5 and the maximum

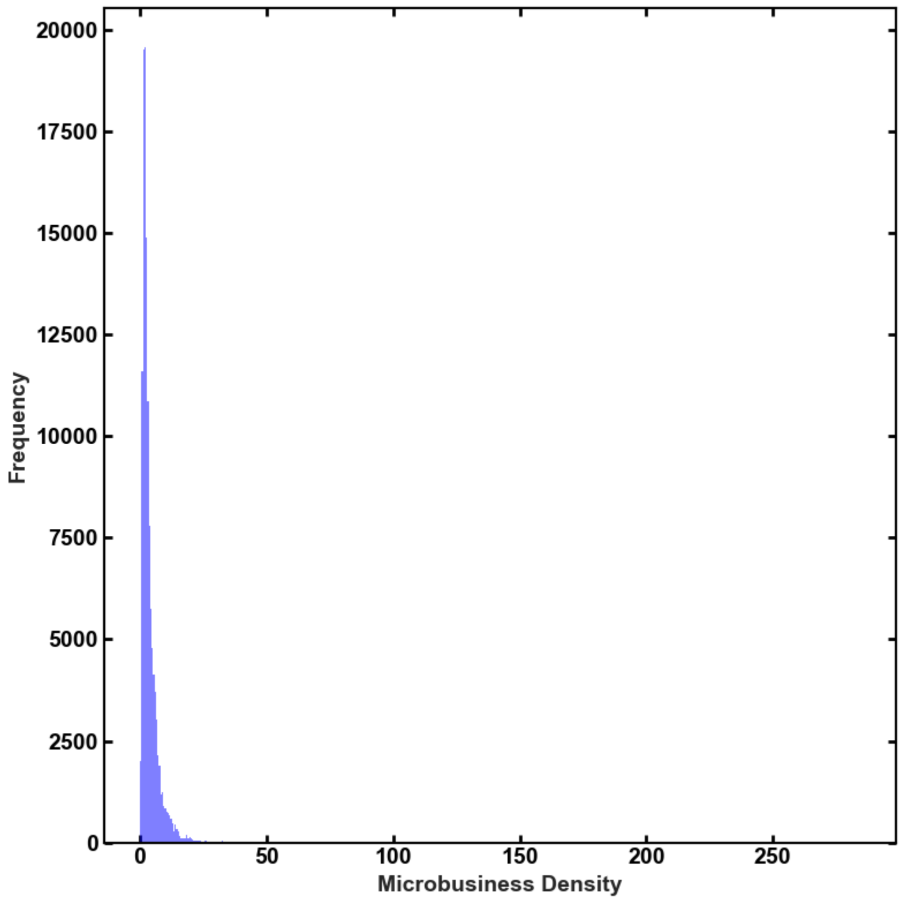
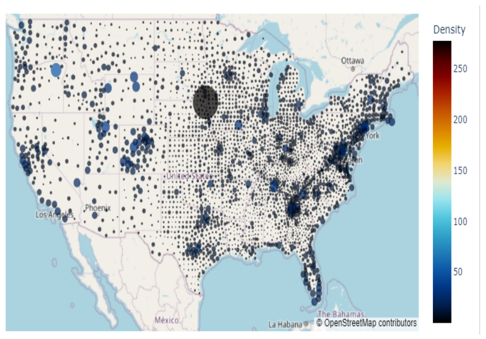


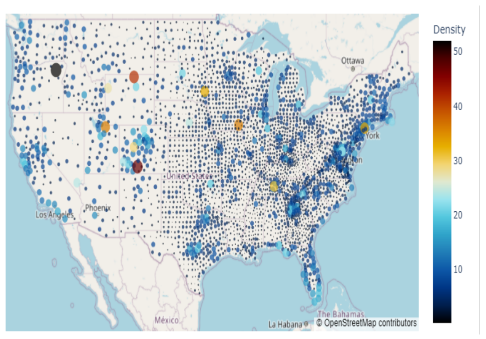
Figure 4: A distribution plot showing the density of microbusinesses of United States.

value a little above 220.

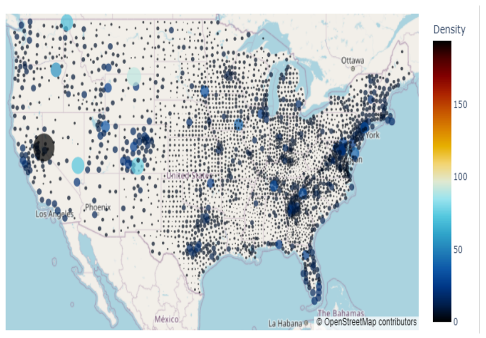
Figure 5 shows the annual changes in the microbusiness density all over the United States, to the county level. It is clear that with the change of time, there are precise interactions between and have a significant increase in the number of businesses. The data have such temporal snapshots of data for 39 months, with 3135 individual time series. This explains why it is essential to bring relationship-based time series forecasting.



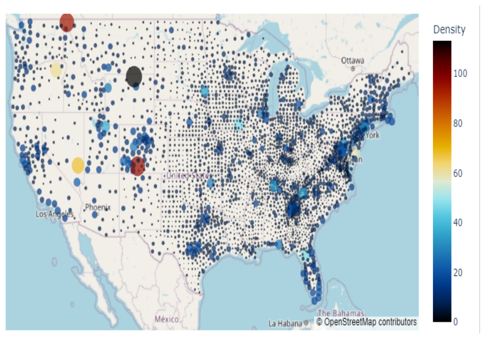
(a) Microbusiness Density Distribution on 2019-08-01



(b) Microbusiness Density Distribution on 2020-08-01



(c) Microbusiness Density Distribution on 2021-08-01



(d) Microbusiness Density Distribution on 2022-08-01

Figure 5: Comparison of Microbusiness Density in August 2019-2022 across the United States. Each subplot displays the microbusiness density distribution for the corresponding year, illustrating the variation in the spatial distribution of micro businesses over time.

3.2.2 Data Preprocessing

Data preprocessing is crucial for time series forecasting, as the predictions are highly sensitive to outliers, data scales, and null values. In addition to that, features engineering is one of the most important things to pay attention. In this study, two different approaches are compared, one is the traditional time series forecasting, and the other is the temporal graph network based time series forecasting. Therefore, the data processing has to be different for both approaches.

3.2.2.1 Data Cleaning and Standardizing

This step aims to transform raw data into a form easily fed into a forecasting model by cleaning, changing, and preparing it for analysis while minimizing the impact of noise, outliers, and missing values. One of the first steps is to check for and handle missing values, outliers, and anomalies. Missing values can be filled using interpolation methods, such as a forward or backward filling or more sophisticated imputation methods. Outliers and monsters can be detected using statistical methods, such as Z-score, IQR, and boxplot. They can be handled by removing or replacing them with more reasonable values.

Another vital aspect is feature engineering, which involves creating new features from the existing ones to capture more information and patterns in the data. This can include computing rolling statistics, such as moving averages, exponential smoothing, lagged variables, or creating dummy variables for categorical variables. Normalization is also a critical step in data preprocessing, as it helps to standardize the scale of the data so that different variables are comparable and have equal weight in the forecasting model. This is particularly important when the variables in the time series data have different ranges or units of measurement. Popular normalization methods include min-max scaling, z-score normalization, and decimal scaling.

3.2.2.2 Temporal Data Preparation

This study prepared the temporal data in a specific format for traditional time series forecasting. The data was divided into time windows for each time step, which allowed the model to be trained on a sliding window basis. This method of preparing the data is important because it considers the temporal nature of the data, ensuring that the model can learn the patterns and trends that occur over time. The time windows also allow the model to be trained on a rolling basis, which means the model is constantly updated as new data becomes available. This ensures the model remains accurate and up-to-date, even as the underlying data changes. Overall, the temporal data preparation process is an essential step in traditional time series forecasting. It plays a crucial role in ensuring the accuracy and reliability of the model's predictions.

3.2.2.3 Graph Data Preparation

Graph data preparation is a critical step in machine learning tasks involving graph structures, such as graph classification, node classification, and link prediction. In this step, the graph data is typically transformed into a numerical representation that can be fed into machine learning models. One common approach for graph data preparation is to use graph embedding methods, which map nodes or subgraphs into low-dimensional

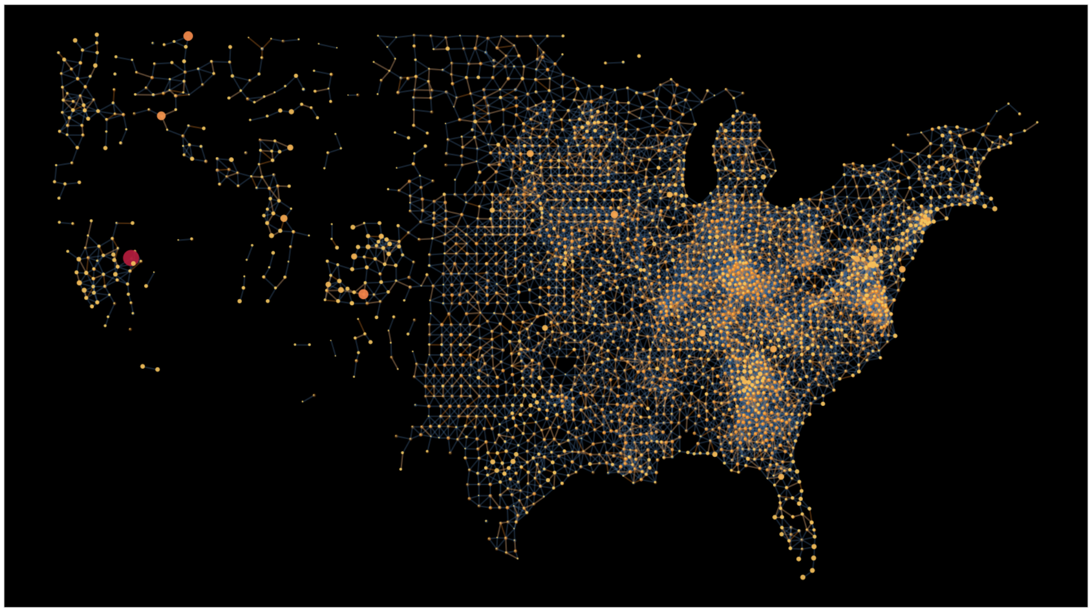


Figure 6: Network Graph of United States with Distance Threshold of 50mile between county to county. Nodes are the counties, and edges are the distance between them.

vector representations. Another approach is to use graph kernels, which measure the similarity between pairs of graphs by comparing their structures. Additionally, various preprocessing techniques, such as node feature normalization, feature selection, and data augmentation, can be applied to improve the quality and efficiency of the input graph data. Figure 6 shows the geospatial representation of the network graph of the united states with counties as nodes, distances as edges, and micro business density values as the node size. An interactive graph visualization dashboard for the processed data is deployed to the Tableau public server [44].

3.3 Experiment Setting

In this section, the experiment setting of the study was discussed, including data partition into training, validation, and test data. The training set is used to train the model; the validation set is used to tune the model's hyperparameters and evaluate its performance during the training. Also, model architectures and hyperparameters for temporal graph neural networks were discussed.

3.3.1 Software and Hardware Stack

3.3.1.1 Software Stack

This study employed a specific software stack to conduct the research and implement the proposed methodology. The software stack consisted of the following tools and libraries:

- **Python Programming Language:** Python was chosen as the primary programming language for its versatility and extensive ecosystem of libraries. Python provided a robust foundation for implementing the various components of the research project.
- **PyTorch Deep Learning Framework:**PyTorch, a widely used deep learning framework, was utilized for developing and training the neural network models. PyTorch offers a flexible and intuitive interface for designing and implementing deep learning architectures, making it suitable for the research objectives.

- **PyTorch Geometric Temporal:** PyTorch Geometric Temporal is a specialized library that extends PyTorch Geometric for temporal graph operations and network training. It provides tools and utilities to handle temporal graph data and train graph neural networks specifically designed for temporal data analysis.
- **NetworkX:**The NetworkX library was employed for graph construction and manipulation. NetworkX offers a comprehensive set of functionalities for creating, analyzing, and modifying graphs, enabling the construction and preprocessing of complex network structures in the research project.
- **Tableau Public Software:** Tableau Public Software was used for creating interactive dashboards and generating reports. It allows for visually appealing and intuitive data visualization, facilitating the effective communication of research findings and insights to stakeholders and readers.
- **Pandas and NumPy:** Pandas and NumPy were utilized for data manipulation and processing. These libraries offer efficient data structures and functions for handling large datasets, enabling data cleaning, transformation, and aggregation operations.

By leveraging this software stack, the research project benefited from the strengths and capabilities of each tool, facilitating the development, training, and evaluation of the proposed methodology for temporal graph analysis.

3.3.1.2 Hardware Stack

The hardware stack employed for this project showcased a formidable configuration capable of handling demanding computational tasks. It consisted of dual AMD EPYC 7H12 CPUs, providing 128 cores. This substantial processing power enabled the efficient execution of computationally intensive operations, such as deep learning training and large-scale data processing. Accompanying the powerful CPUs was an impressive 512GB of RAM, ensuring ample memory capacity to handle large datasets and complex computations. Including eight Nvidia RTX A6000 GPUs, each equipped with 48GB of RAM, further bolstered the system's capabilities. These high-end GPUs offered exceptional parallel processing capabilities, specifically accelerating deep learning tasks and training neural network models. The hardware stack was also equipped with 3.5TB of local scratch space, specifically designated for storing temporary files and intermediate results. This extensive storage capacity contributed to efficiently managing and processing transient data throughout the project's execution.

3.3.2 Framework Design and Comparison Models

In this section, the model design for this study is discussed. This whole study used two different approaches to design the experiment of time series forecasting. The first approach uses statistical and machine learning models; the second is temporal graph neural networks. The idea is to show the performance of both techniques, compare the results with both design pipelines, and validate the hypothesis.

3.3.2.1 Statistical and Machine Learning Models

For statistical methods, SARIMA and Holts Winters Exponential seasoning are used. For SARIMA, the order of (0,1,1) and seasonal order of (1,1,0,2) are the optimized values, whereas, for the Holt winters exponential seasoning, a grid search-based hyper-parameter search was conducted; for machine learning models, linear regression, random forest, and support vector machine-based regressors are used. Data was unshuffled to keep the temporal and lag information with the lagging technique.

3.3.2.2 Temporal Graph Neural Networks

For the Temporal graph neural networks, attention temporal graph convolutional network (A3T-GCN) [39] architecture is used for the core layer, which is passed through the final linear layer. In this study, only a single layer of an A3T-GCN block is used to avoid the efficiency issues with higher information condensation and bottlenecking with increased TGCN blocks.

The T-GCN model combines Graph Convolutional Networks (GCN) and Gated Recurrent Units (GRU) to capture spatiotemporal characteristics of temporal graph data. Given n historical time series traffic data, the T-GCN model produces n hidden states (h) that cover spatiotemporal features: h(t-n), ..., h(t-1), h(t). The GCN layer first processes the input data to capture the spatial dependencies of the data. The output of the GCN layer is then fed into the GRU layer to model the temporal dependencies. The update gate (ut), reset gate (rt), and candidate memory cell (ct) are calculated as follows:

$$u_t = \sigma(W_u * (GC(A, X_t), h_{t-1})) \tag{3.1}$$

$$r_t = \sigma(W_r * (GC(A, X_t), h_{t-1})) \tag{3.2}$$

$$c_t = tanh(W_c * (GC(A, X_t), (r_t * h_{t-1}))) \tag{3.3}$$

$$h_t = u_t * h_{t-1} + (1 - u_t) * c_t \tag{3.4}$$

The hidden states are then fed into an attention model to determine the context vector covering global traffic variation information. The weight of each hidden state is calculated by Softmax using a multilayer perception: a(t-n), ..., a(t-1), a(t). The weighted sum calculates the context vector that covers the global traffic variation information:

$$c = \sum_{i=t-n}^{t} a_i h_i \tag{3.5}$$

where a_i is the weight of the ith hidden state.

The A3TGCN is an extension of the TGCN model by adding an attention mechanism. Over time, the attention mechanism computes the attention weights $\alpha(t{-}n), ..., \alpha(t{-}1), \alpha(t)$ for each hidden historical state $h(t{-}n), ..., h(t{-}1), h(t)$ obtained from the TGCN model, based on its relevance to the current time step t. The context vector $c(t)$ that captures the global traffic variation at time t is then obtained by taking the weighted sum of the hidden historical states, where the attention weights serve as the weights for the weighted sum. The attention weights $\alpha(t-n), ..., \alpha(t-1), \alpha(t)$ are computed using a two-layer feedforward neural network that takes the current hidden state $h(t)$ as input. The output of the neural network is then passed through a Softmax function to obtain the attention weights:

$$\alpha(t-i) = \frac{\exp(f(h(t), h(t-i)))}{\sum_{j=n}^{t-1} \exp(f(h(t), h(t-j)))} \tag{3.6}$$

Where $f(h(t), h(t-i))$ is the score function that measures the similarity between the current hidden state $h(t)$ and the hidden historical state $h(t-i)$. The context vector $c(t)$ is then computed as follows:

$$c(t) = \sum_{i=n}^{t} \alpha(t-i) * h(t-i) \tag{3.7}$$

The context vector $c(t)$ is then concatenated with the current hidden state $h(t)$ and fed into the output layer to produce the final output of the A3TGCN model.

3.3.3 Framework Validation

The current Attention-based Temporal Graph Convolutional Network was validated with the benchmark datasets, which are very similar in structure and problem definition. Two data sets, SZ - Taxi Dataset and Los - Loop Dataset [39], are used to validate the framework.

3.3.3.1 Data Description

The study utilized two real-world traffic datasets: the SZtaxi dataset, which consists of taxi trajectory data in Shenzhen City, and the Losloop detector dataset, which contains traffic speed data collected from loop detectors on highways in Los Angeles County. The experiments focused on analyzing traffic speed as the primary traffic information. The SZtaxi dataset covers the taxi trajectory data in Shenzhen from January 1 to January 31, 2015, specifically focusing on 156 significant roads in the Luohu District. On the other hand, the Losloop dataset comprises real-time traffic speed data collected from 207 sensors along the highway in Los Angeles County from March 1 to March 7, 2012.

3.3.3.2 Evaluation Metrics and Validation Approach

To evaluate the prediction performance of the model, the error between the actual traffic speed and predicted results is assessed based on the following metrics:

Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{1}{MN}\sum_{j=1}^{N}\sum_{i=1}^{M}(y_{ji} - \hat{y}_{ji})^2} \tag{3.8}$$

Mean Absolute Error (MAE):

$$MAE = \frac{1}{MN}\sum_{j=1}^{N}\sum_{i=1}^{M}|y_{ji} - \hat{y}_{ji}| \tag{3.9}$$

Accuracy:

$$Accuracy = 1 - \frac{Y - YF}{YF} \tag{3.10}$$

Coefficient of Determination (R2):

$$R2 = 1 - \frac{\sum_{j=1}^{N}\sum_{i=1}^{M}(y_{ji} - \hat{y}ji)^2}{\sum j = 1^N \sum_{i=1}^{M}(y_{ji} - \bar{Y})^2} \tag{3.11}$$

Explained Variance Score (var):

$$var = 1 - \frac{Var(Y - \hat{Y})}{Var(Y)} \tag{3.12}$$

In these formulas, y_{ji} and $\hat{y}ji$ represent the real and predicted traffic information of temporal sample j on the road i, respectively. N is the number of nodes on the road, and M is the number of temporal samples. Y and YF are the sets of yji and $\hat{y}_{ji}$, respectively, and $\bar{Y}$ is the mean of Y. Various models with the historical average model(HA), auto-regressive integrated moving average model(ARIMA), SVR, GCN model, and GRU model are trained and compared with the AT3TGCN model to validate the framework.

3.3.4 Data Partition

The data collection and a few necessary details about the data set are discussed in the sections above. The main focus of this study is to work with the US micro business density distribution data set, which is available for 39 months from 2019-08 to 2022-10, with one data point for each month and per county. This data have 3135 unique time series for the given time. The training and validation data came from the first 32 months, while the testing data is from the last seven months.

3.3.5 Baselines

Evaluating the performance of time series forecasting models is crucial for determining their effectiveness, and this can be achieved by comparing the model's predictions to the baseline models. In this study, the main baseline models used are naive baseline, seasonal naive, and statistical methods. A more detailed theory was given in Chapter 2.

Naive baseline forecasting involves using the most recent observation as the forecast for the next time step. This simple and easy-to-implement baselin can serve as a benchmark for more complex models. The equation for the model is as follows:

$$\hat{y}_{t+1} = y_t \tag{3.13}$$

where $haty_{t+1}$ is the predicted value for the next time step and y_t is the most recent observation.

The naive seasonal model uses the observation from the same season in the previous year as the forecast for the current time step. This baseline is useful for datasets that exhibit seasonality. The equation for the naive seasonal model is as follows:

$$\hat{y}_{t+1} = y_{t+1-m} \tag{3.14}$$

where $haty_{t+1}$ is the predicted value for the next time step and m is the number of time steps in a season.

The statistical method is discussed in chapter 2. These baseline models provide a reference point for evaluating the performance of more complex models. Metrics such as SMAPE, MAPE, and RMSE can be used to compare the accuracy of the baseline models to the more sophisticated forecasting models.

3.3.6 Evaluation Metrics

When evaluating the performance of a time series forecasting model, it is crucial to use appropriate evaluation metrics. In this study, metrics used for measuring the accuracy of a forecast are Symmetric Mean Absolute Percentage Error (SMAPE), Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE).

- **SMAPE** is a symmetric metric that measures the percentage difference between the actual and predicted values. It is defined as:

$$SMAPE = \frac{100\%}{n}\sum_{t=1}^{n}\frac{|F_t - A_t|}{(|F_t| + |A_t|)/2} \tag{3.15}$$

  where n is the number of observations in the test set, F_t is the forecast value for time t, A_t is the actual value for time t.

- **MAPE** is another commonly used metric that measures the percentage difference between the actual and predicted values, but unlike SMAPE, it is not symmetric. MAPE is defined as:

$$MAPE = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \tag{3.16}$$

  where A_t is the actual value at time t, F_t is the forecasted value at time t, and n is the total number of observations.

- **RMSE** is a metric that measures the root of the mean squared difference between the actual and predicted values. It is defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(F_t - A_t)^2} \tag{3.17}$$

  where n is the number of observations in the dataset, F_t is the forecasted value at time t, A_t is the actual (observed) value at time t

CHAPTER 4

RESULTS AND DISCUSSION

This section presents the results of framework validation with different metrics and benchmark datasets. Later the graph data preparation of U.S. Microbusiness density data was discussed, and a temporal graph neural network model called A3TGCN was used to perform time series forecasting.

In the second experiment, the results of proposed experiments on time series forecasting using traditional methods, machine learning models, and neural graph networks. The first experiment involved building a pipeline for assessing the baseline time series forecasting models with single time series. The model's performance is evaluated with Naive, Seasonal Naive, SARIMA, and Prophet.

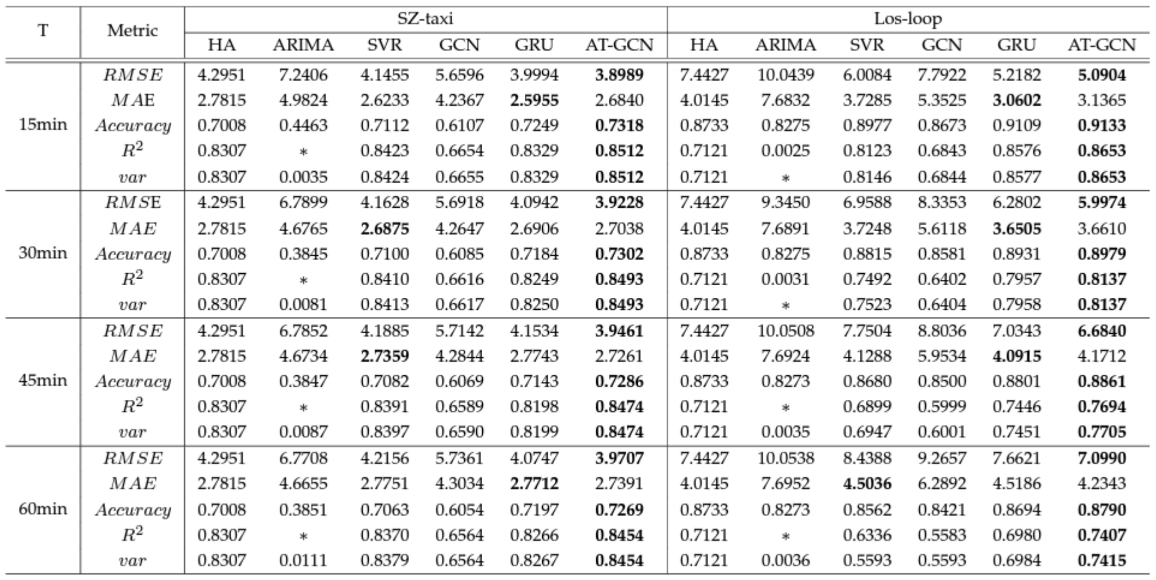
In the last experiment, time series forecasting of multiple time series using the baseline models and added the sum of losses to compare the performance of the models to the advanced temporal graph-based model. This experiment showed that the Naive model is inefficient, while Holt winters smoothing method outperformed the other baseline models.

Finally, the results of the A3TGCN model have been compared its results with the traditional methods and machine learning models for all the counties in Missouri State. Our results showed that the A3TGCN model outperformed all the other models regarding MAPE, SMAPE, and root mean square error (RSME) of individual time series predictions.

4.1 Framework Validation

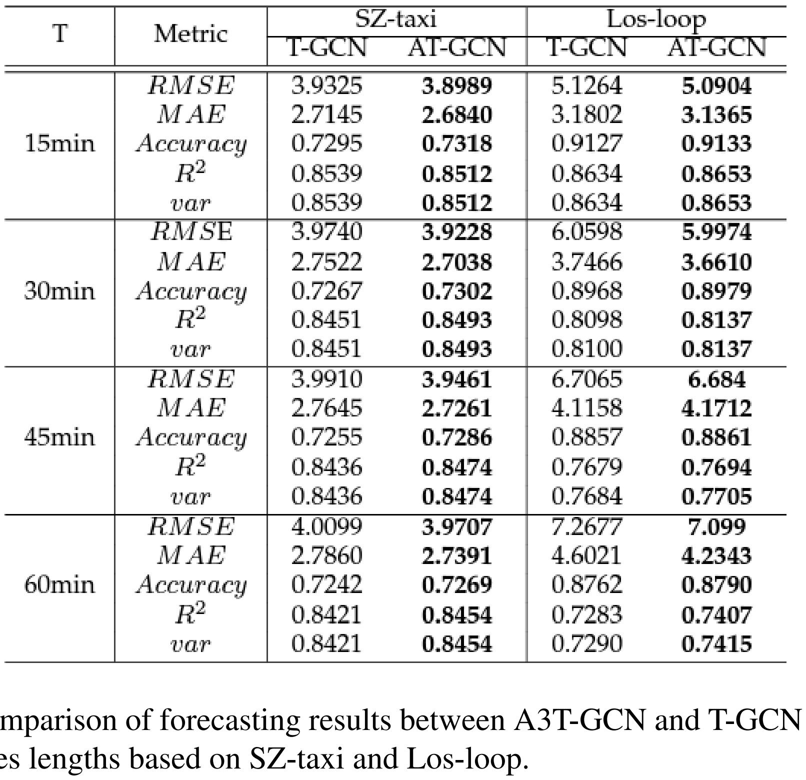
The AT-GCN model's hyperparameters include the learning rate, epoch, and a number of hidden units. The learning rate and epoch were manually set for both datasets based on previous experiences, with values of 0.001 and 5000, respectively. The number of hidden units was 64 for SZtaxi and 100 for Losloop. In this study, 80% of the traffic data was utilized as the training set, while the remaining 20% served as the test set. The model's predictive performance was evaluated by comparing the predicted results for the next 15, 30, 45, and 60 minutes with those generated by the historical average model (HA), auto-regressive integrated moving average model (ARIMA), support vector regression (SVR), GCN model, and GRU model. The AT-GCN model was analyzed from several perspectives: precision, spatiotemporal prediction capabilities, long-term prediction capability, and global feature capturing capability.

Figure 7 [39] shows the performance of the TGCN model with traditional models, and it is clear that TGCN performed the conventional models, whereas figure 8 shows that Attention-based TGCN is over performing the conventional TGCN. This Validated the framework with metrics evaluation. Figure 9, and Figure 10 show the visualization results for a prediction horizon of 15,30,45,60mins for both the SZ-taxi and LOS-loop datasets. These results validate the Framework and its use for similar applications.



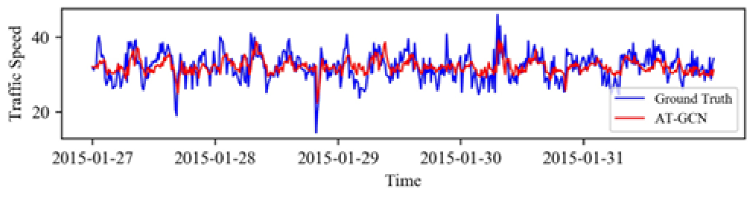
T	Metric	SZ-taxi						Los-loop					
		HA	ARIMA	SVR	GCN	GRU	AT-GCN	HA	ARIMA	SVR	GCN	GRU	AT-GCN
	$RMSE$	4.2951	7.2406	4.1455	5.6596	3.9994	**3.8989**	7.4427	10.0439	6.0084	7.7922	5.2182	**5.0904**
	MAE	2.7815	4.9824	2.6233	4.2367	**2.5955**	2.6840	4.0145	7.6832	3.7285	5.3525	**3.0602**	3.1365
15min	$Accuracy$	0.7008	0.4463	0.7112	0.6107	0.7249	**0.7318**	0.8733	0.8275	0.8977	0.8673	0.9109	**0.9133**
	R^2	0.8307	*	0.8423	0.6654	0.8329	**0.8512**	0.7121	0.0025	0.8123	0.6843	0.8576	**0.8653**
	var	0.8307	0.0035	0.8424	0.6655	0.8329	**0.8512**	0.7121	*	0.8146	0.6844	0.8577	**0.8653**
	$RMSE$	4.2951	6.7899	4.1628	5.6918	4.0942	**3.9228**	7.4427	9.3450	6.9588	8.3353	6.2802	**5.9974**
	MAE	2.7815	4.6765	**2.6875**	4.2647	2.6906	2.7038	4.0145	7.6891	3.7248	5.6118	**3.6505**	3.6610
30min	$Accuracy$	0.7008	0.3845	0.7100	0.6085	0.7184	**0.7302**	0.8733	0.8275	0.8815	0.8581	0.8931	**0.8979**
	R^2	0.8307	*	0.8410	0.6616	0.8249	**0.8493**	0.7121	0.0031	0.7492	0.6402	0.7957	**0.8137**
	var	0.8307	0.0081	0.8413	0.6617	0.8250	**0.8493**	0.7121	*	0.7523	0.6404	0.7958	**0.8137**
	$RMSE$	4.2951	6.7852	4.1885	5.7142	4.1534	**3.9461**	7.4427	10.0508	7.7504	8.8036	7.0343	**6.6840**
	MAE	2.7815	4.6734	**2.7359**	4.2844	2.7743	2.7261	4.0145	7.6924	4.1288	5.9534	**4.0915**	4.1712
45min	$Accuracy$	0.7008	0.3847	0.7082	0.6069	0.7143	**0.7286**	0.8733	0.8273	0.8680	0.8500	0.8801	**0.8861**
	R^2	0.8307	*	0.8391	0.6589	0.8198	**0.8474**	0.7121	*	0.6899	0.5999	0.7446	**0.7694**
	var	0.8307	0.0087	0.8397	0.6590	0.8199	**0.8474**	0.7121	0.0035	0.6947	0.6001	0.7451	**0.7705**
	$RMSE$	4.2951	6.7708	4.2156	5.7361	4.0747	**3.9707**	7.4427	10.0538	8.4388	9.2657	7.6621	**7.0990**
	MAE	2.7815	4.6655	2.7751	4.3034	**2.7712**	2.7391	4.0145	7.6952	**4.5036**	6.2892	4.5186	4.2343
60min	$Accuracy$	0.7008	0.3851	0.7063	0.6054	0.7197	**0.7269**	0.8733	0.8273	0.8562	0.8421	0.8694	**0.8790**
	R^2	0.8307	*	0.8370	0.6564	0.8266	**0.8454**	0.7121	*	0.6336	0.5583	0.6980	**0.7407**
	var	0.8307	0.0111	0.8379	0.6564	0.8267	**0.8454**	0.7121	0.0036	0.5593	0.5593	0.6984	**0.7415**

Figure 7: Comparison of prediction results between the T-GCN model and other baseline methods on SZ-taxi and Los-loop datasets.

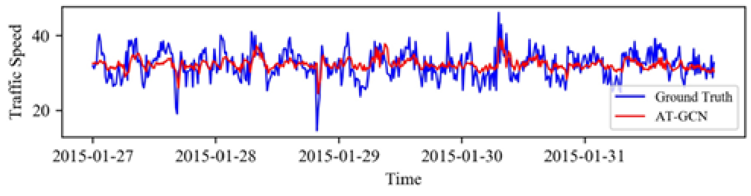


T	Metric	SZ-taxi		Los-loop	
		T-GCN	AT-GCN	T-GCN	AT-GCN
	$RMSE$	3.9325	**3.8989**	5.1264	**5.0904**
	MAE	2.7145	**2.6840**	3.1802	**3.1365**
15min	$Accuracy$	0.7295	**0.7318**	0.9127	**0.9133**
	R^2	0.8539	**0.8512**	0.8634	**0.8653**
	var	0.8539	**0.8512**	0.8634	**0.8653**
	$RMSE$	3.9740	**3.9228**	6.0598	**5.9974**
	MAE	2.7522	**2.7038**	3.7466	**3.6610**
30min	$Accuracy$	0.7267	**0.7302**	0.8968	**0.8979**
	R^2	0.8451	**0.8493**	0.8098	**0.8137**
	var	0.8451	**0.8493**	0.8100	**0.8137**
	$RMSE$	3.9910	**3.9461**	6.7065	**6.684**
	MAE	2.7645	**2.7261**	4.1158	**4.1712**
45min	$Accuracy$	0.7255	**0.7286**	0.8857	**0.8861**
	R^2	0.8436	**0.8474**	0.7679	**0.7694**
	var	0.8436	**0.8474**	0.7684	**0.7705**
	$RMSE$	4.0099	**3.9707**	7.2677	**7.099**
	MAE	2.7860	**2.7391**	4.6021	**4.2343**
60min	$Accuracy$	0.7242	**0.7269**	0.8762	**0.8790**
	R^2	0.8421	**0.8454**	0.7283	**0.7407**
	var	0.8421	**0.8454**	0.7290	**0.7415**

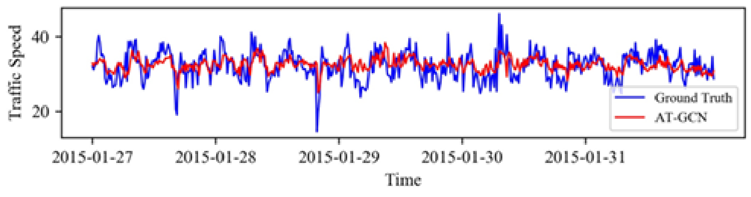
Figure 8: Comparison of forecasting results between A3T-GCN and T-GCN under different time series lengths based on SZ-taxi and Los-loop.



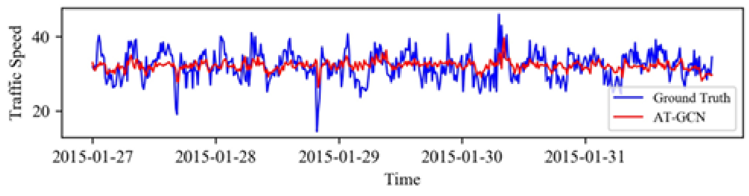
(a) 15 minutes



(b) 30 minutes

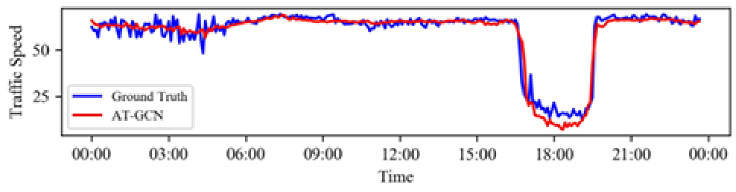


(c) 45 minutes

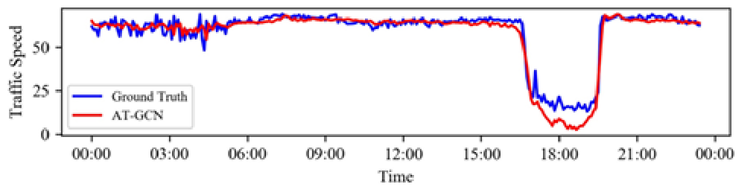


(d) 60 minutes

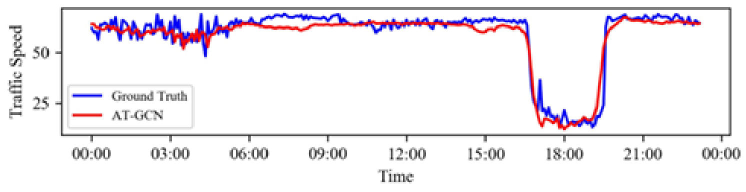
Figure 9: The visualization results for a prediction horizon of 15,30,45,60 minutes(SZ-taxi).



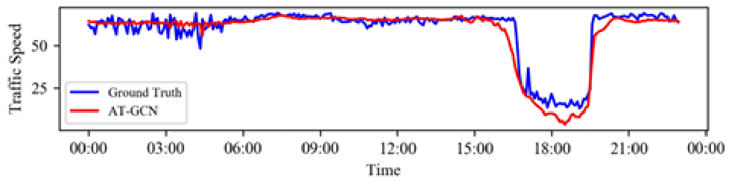
(a) 15 minutes



(b) 30 minutes



(c) 45 minutes



(d) 60 minutes

Figure 10: The visualization results for a prediction horizon of 15,30,45,60 minutes(Los-loop).

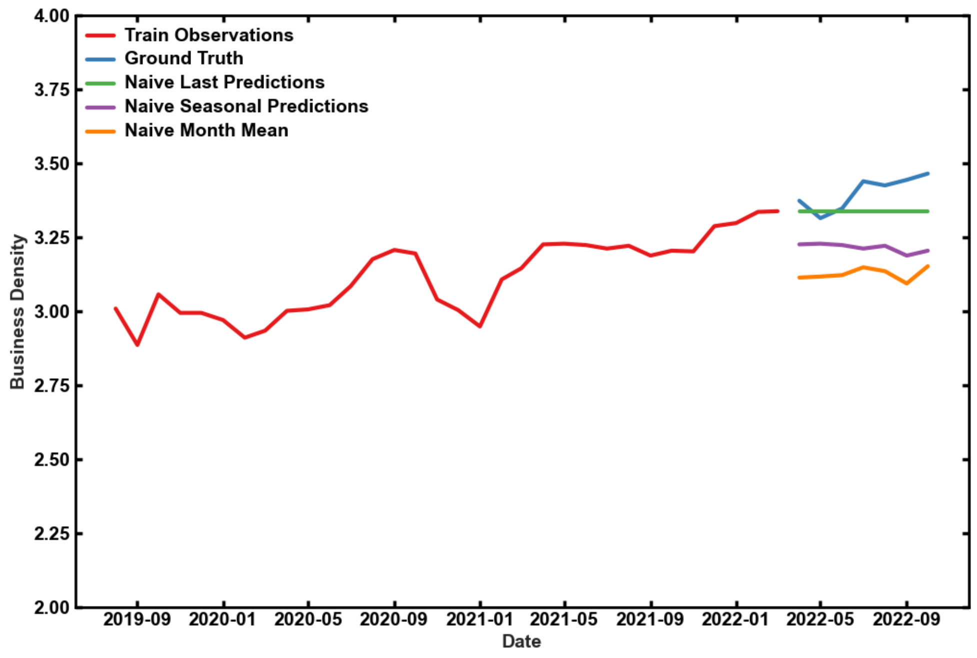


Figure 11: Plot of training data and baseline predictions using naive, seasonal naive, and month mean naive methods of microbusiness density a single time series. The red line represents the actual training data, while the green, purple, and blue lines represent the predictions of the naive, seasonal naive, and month mean naive methods, respectively. It can be observed that the baseline methods do not capture the seasonality and trends in the training data quite well and, therefore, perform poorly in making accurate predictions.

4.2 Experiment - 1: Single Time Series Forecasting

This experiment developed single time series forecasting models using various baseline models such as Last Value Naive, Seasonal Naive, Month Mean Naive, Holt-Winters Exponential Smoothing, SARIMA, Prophet, Random Forest, Linear Regression, and Support Vector Machine. The detailed theory was discussed in Chapter 3; the initial baseline is the last naive, seasonal naive, and month mean naive methods. Figure 11 shows the baseline predictions of naive baseline methods, including naive, seasonal naive, and month mean naive methods. It is evident from the plot that the naive, seasonal naive, and month mean naive baseline methods do not effectively capture the patterns of seasonality and trends present in the training data. Hence, their performance in generating accurate predictions is poor. Therefore it is reasonable to train the model with other machine learning models and statistical learning methods.

Figure 12 shows the predictions of microbusiness density of a single time series with SARIMA, Holt-Winters smoothing, Prophet for the last 7 months of data, and the training data. With visual observation, it is clear that statistical methods are following the trend pattern compared to Figure 11. Through this experiment, it is evident that Holt-Winters performed the best, while Prophet performed the worst. The poor performance of the Prophet model may be because the data analyzed in this experiment was at a monthly level. In contrast, Prophet is typically designed to work with data daily and considers holiday effects.

Figure 13 shows the predictions of microbusiness density of a single time series

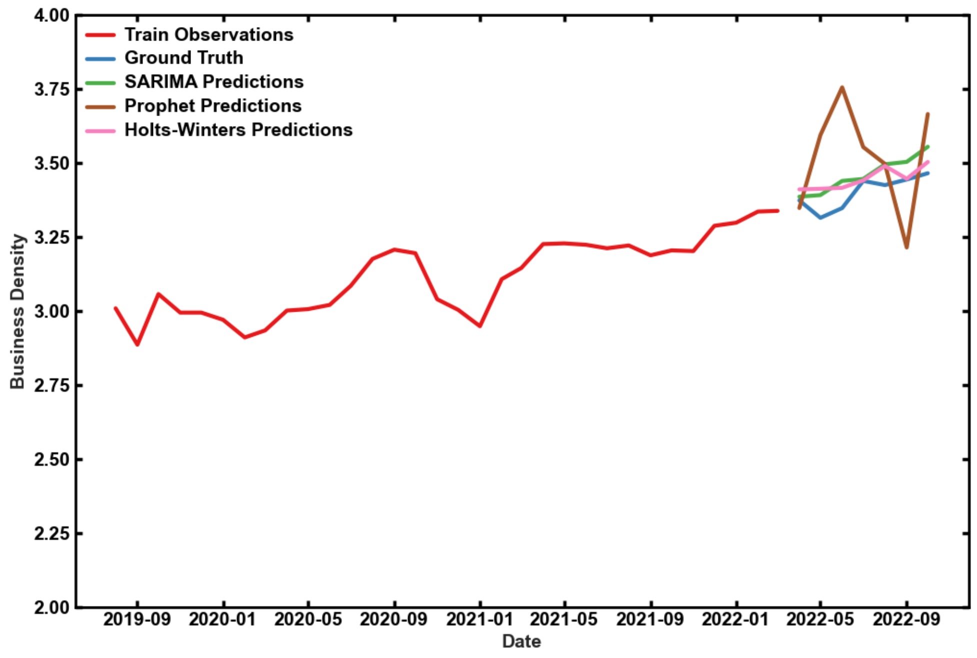


Figure 12: Plot of training data and the predictions made by the grid-searched Holt-Winters, SARIMA, and Prophet models, for a single time series. All the models have captured the seasonality and trends in the training data quite well and have made accurate predictions. Holt-Winters performed the best, while Prophet performed the worst.

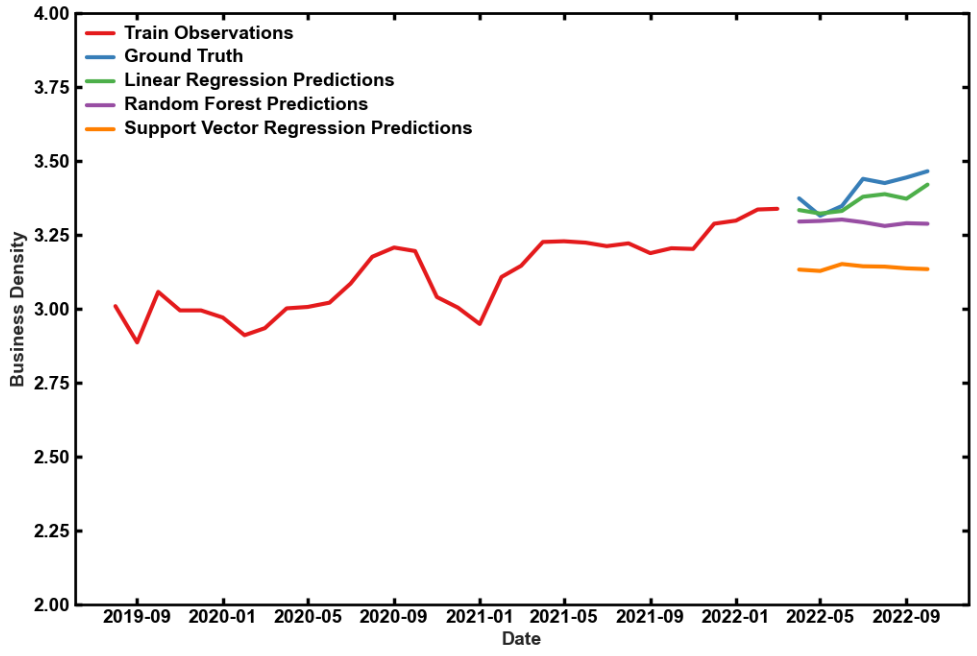


Figure 13: Plot of training data and the predictions made by linear regression, random forest regressor, and support vector regressor for a single time series. These models did not capture the seasonality and trends in the training data.

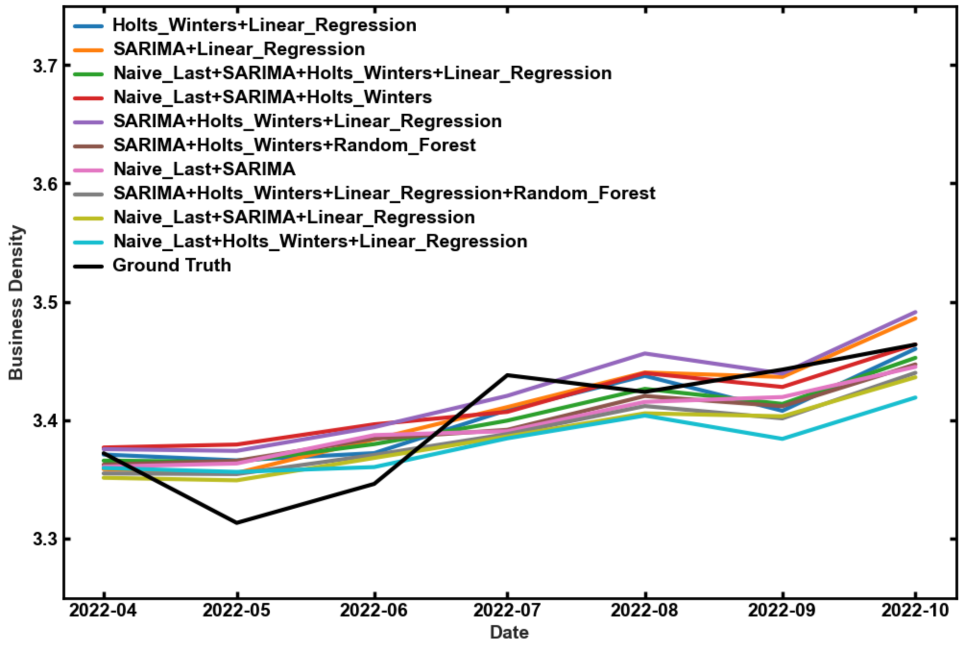


Figure 14: Plot of Ground Truth and the ensemble predictions with various baseline models. All the ensembles captured the trend of the test data.

with machine learning models, including linear regression, random forest, support vector regression for the last 7 months of data, and the training data. These models did not capture the trend and seasonality well compared to the statistical methods. The overall performance of these baselines is compared with Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), and Symmetric Mean Absolute Percentage Error (SMAPE) in Table 2, from this linear table regression, performed exceptionally well compared to the other models. With the predictions of these models, ensembles of all possible combinations are created to improve the performance of the time series forecasting; the ensemble performed here is the mean average of the predictions.

Figure 14 shows the top 10 ensemble combinations, and most of the predictions

Table 2: Comparison of Forecasting Models

Forecast	RMSE	MAPE	SMAPE
Linear Regression	0.04	1.16	1.17
Holt-Winters	0.06	1.32	1.30
SARIMA	0.07	1.71	1.69
Last Naive	0.08	2.04	2.07
Random Forest	0.12	3.19	3.26
Prophet	0.23	5.60	5.45
Season Naive	0.20	5.46	5.63
SVM	0.27	7.71	8.03
Month Mean Naive	0.28	8.07	8.42

Table 3: Comparison of Ensemble Forecasting Models

Ensemble	RMSE	MAPE	SMAPE
Holts-Winters, Linear Regression	0.03	0.68	0.34
SARIMA, Linear Regression	0.03	0.70	0.35
Naive Last, SARIMA, Holts-Winters, Linear Regression	0.03	0.73	0.36
Naive Last, SARIMA, Holts-Winters	0.03	0.78	0.39
SARIMA, Holts-Winters, Linear Regression	0.03	0.82	0.41
SARIMA, Holts-Winters, Random Forest	0.03	0.83	0.42
Naive Last, SARIMA	0.03	0.85	0.42
SARIMA, Holts-Winters, Linear Regression + Random Forest	0.03	0.88	0.44
Naive Last, SARIMA, Linear Regression	0.03	0.90	0.45
Naive Last, Holts-Winters, Linear Regression	0.04	1.03	0.52

capture the trend o the ground truth value. Table 3 shows the top ten ensemble combinations out of 200 plus varieties with the forecasts in hand. From this, holts-winters smoothing and linear regression combination outperformed all others. From this experiment, it is clear that statistical methods are still best for time series forecasting, but what if multiple time series are related somehow? This question needs further experimentation and is covered in the following experiments.

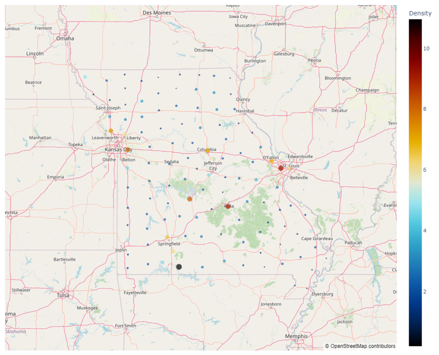
4.3 Experiment - 2: Multiple Time Series Forecasting

In this section, a group of individual time series is trained with all the baseline models from experiment -1, and the performance of the unique time series is evaluated with the SMAPE, MAPE, and RMSE metrics. The idea is to select a group of time series data that belong to the U.S. microbusiness density dataset and implement the proposed methodology to validate the individual time series. For this experimentation, county-level time series of Missouri are selected, and there are 115 counties in this state, which means there are 115 individual time series present in the group for 39 months. Figure 15 shows the microbusiness density at six months intervals throughout 2019-2022; from this figure, one key observation made is that counties near the cities have the highest business density in that area. The training of each county-wise time-series is performed in the first 32 months, and the last seven months are hold of the testing, and a rolling validation is performed with the train data.

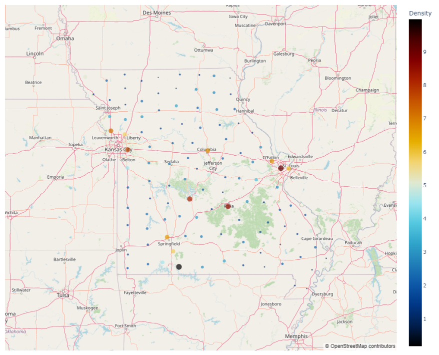
Table 4: Comparison of Missouri State Level Forecasting Metrics with Baseline

Forecast	RMSE	MAPE	SMAPE
holt_winters	0.10	2.59	2.66
last_naive	0.10	2.68	2.80
random_forest	0.14	4.33	4.28
season_naive	0.17	5.56	5.77
sarima	0.18	5.83	6.01
linear_regression	0.19	6.49	6.70
svm	0.21	6.68	7.08
month_mean_naive	0.25	8.12	8.10
prophet	0.43	15.56	14.36

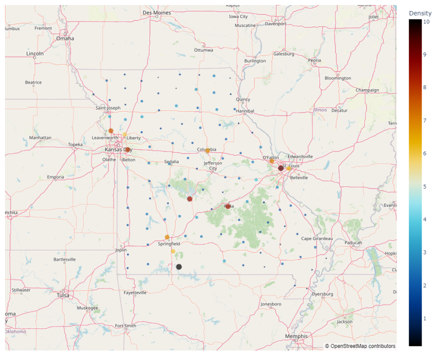
Table 4 shows the testing results of the cumulative sum of errors for each county



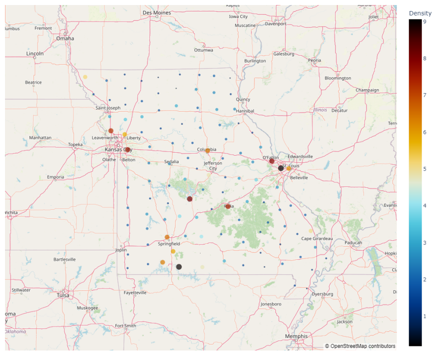
(a) Microbusiness Density Distribution on 2019-08-01



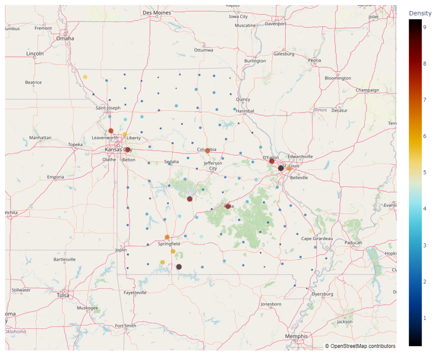
(b) Microbusiness Density Distribution on 2020-02-01



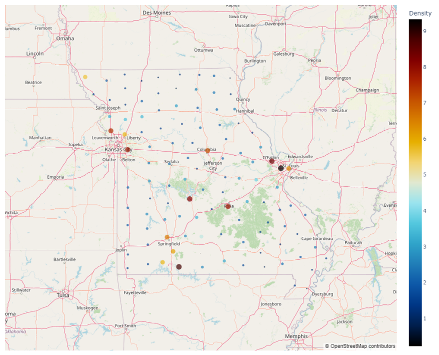
(c) Microbusiness Density Distribution on 2020-08-01



(d) Microbusiness Density Distribution on 2021-02-01



(e) Microbusiness Density Distribution on 2021-08-01



(f) Microbusiness Density Distribution on 2022-02-01

Figure 15: Comparison of Microbusiness Density of Missouri State for the Years 2019-2022 for every six months. Each subplot displays the microbusiness density distribution for the corresponding year, illustrating the variation in the spatial distribution of micro businesses over time.

in the Missouri state business density forecasting. Overall, the Holt winters total errors are lesser than the other models, and Facebook's prophet performed poorly due to the reasoning given in experiment 1. The ensemble creation of this level of forecasting is quite impractical due to the significantly massive number of possible combinations, so only these forecast metrics are considered as the baseline for experiment 3, where the same forecast problem is aimed to tackle by taking advantage of the relationships between the counties, like distance, population density, or something similar.

4.4 Experiment - 3: Temporal Graph Neural Networks

In this section, the most crucial part of this research was discussed. The formulation of the graph problem is unique due to the assumptions that need to make at every stage. Unlike experiment -2, here, all the time series are treated as one whole spatial network that changes with time in a temporal dimension, i.e., a group's time series data is somehow interdependent and has hidden relations.

4.4.1 Creation of the Temporal Snapshots of Graph States

In spatial-temporal graph neural networks, the first and most important task is to convert the given problem into a graph problem with the temporal steps. i.e., all the structured or non-structured information needs to convert into vertices and edge mapping for each time step. In this step, we define the graph variables and the relations based on the type of prediction we are trying to solve. The graph-based networks are usually helpful in node-level and edge-level predictions, and the features of the two-level predictions can be any relevant features. The node and edge could be anything of interest with some connectivity or tangible relation [45–47], and the features could be any averages or moving parameters.

In this study, the U.S. Microbusineess density data have geographical location-based connectivity as these are the time series of the counties in the united states, or else the clustering of the counties together based on the population could be another way of establishing a relationship. Therefore it is possible to treat this data as graph data with few assumptions. These assumptions are as follows.

- **Nodes:** In this study, time series forecasting of county-level microbusiness density is the desired task. Therefore, this individual county can be treated as a node, and node features could be the data related to the county. The data considered here are of two types,

    1. **Temporal Features:** It is only the temporal information based on the lags to the future.
    2. **Temporal + County Information:** It is the county-level past census data, including statistics of families and economic and social open information available on the government census database.

- **Edges:** Edges are the connections between the nodes, and intuitively they give the existence of a relationship between nodes and pass the information in the training phase to learn about the neighbor as a whole network. This study determines these

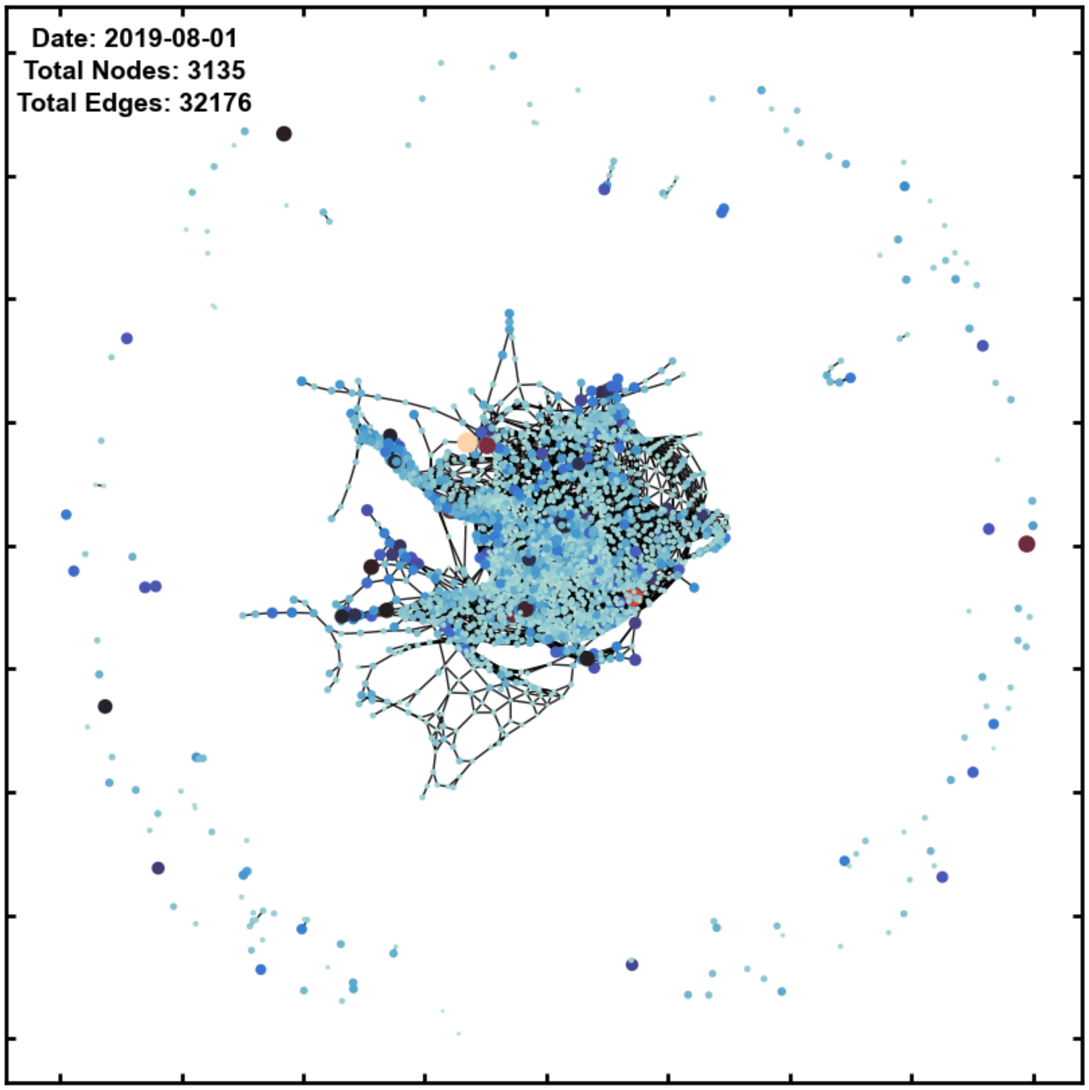


Figure 16: Data visualization of the graph structure on 2019-08-01

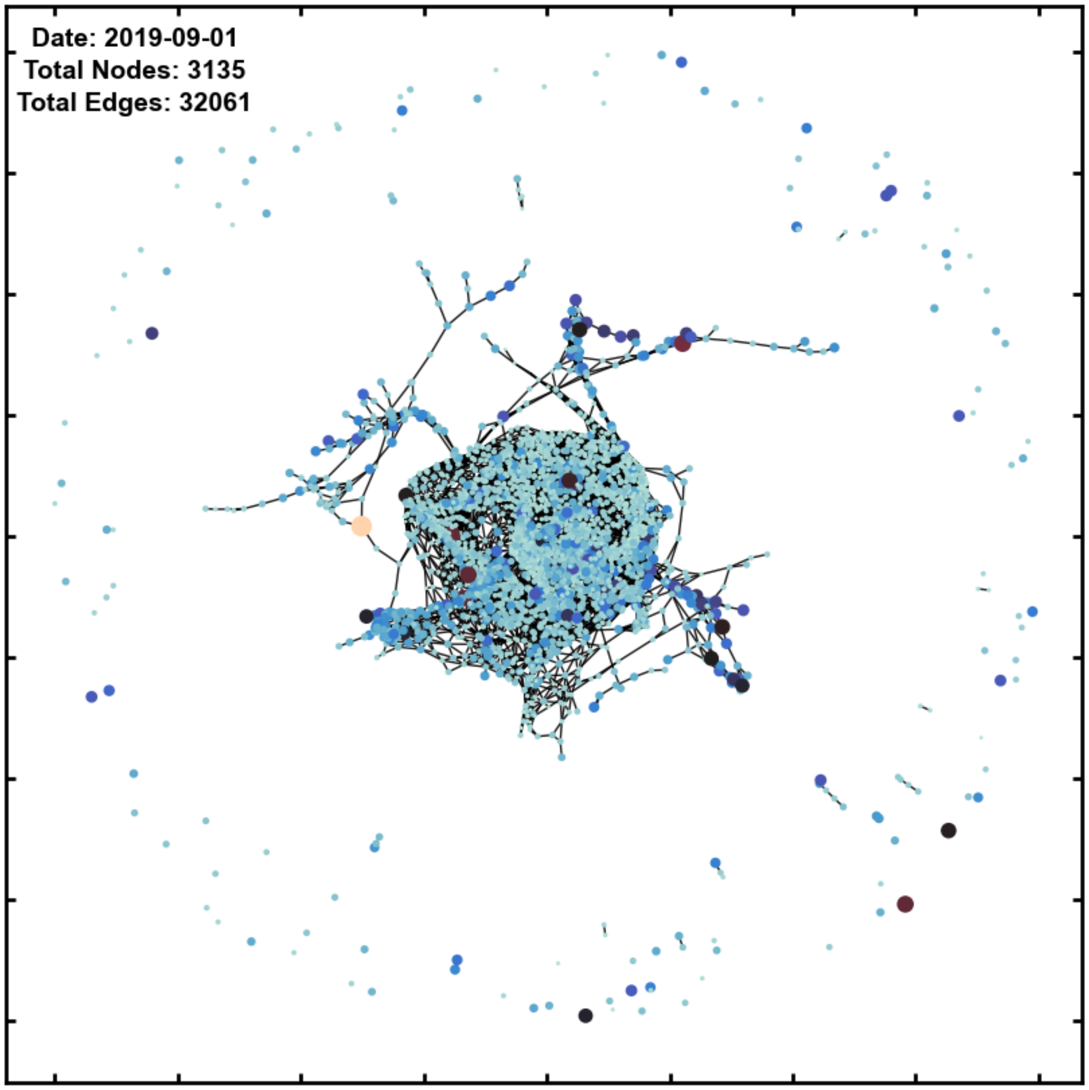


Figure 17: Data visualization of the graph structure on 2019-09-01

edges based on the distance between the nodes. As per the literature on city planning, the impactful growth space between two towns is 50 miles, i.e., when two cities within reach of 50 miles significantly impact each other's growth and dynamic changes within these towns. Hence edge connections were established with a threshold of 50 miles between all the counties. Edge features are called edge weights, and in this study, the connecting distance between the counties is considered the edge feature for the edge. This feature could be anything; for instance, the average travel time could be one potential feature.

As these are temporal graphs, graphs could be dynamic or static. In dynamic graphs, the graph structure changes with each time snapshot; however, in static graphs, this graph structure remains unchanged.

1. **Static Graphs:** In the static graphs, all the possible edges within connection to other counties and under a 50-mile radius are considered.
2. **Dynamic Graphs:** To mimic the dynamic graph structures, a random number of edges are dropped with an edge connection threshold. This creates a robust graph with less overfitting, and the threshold used is 25% of the total number of edges.

Figure 16,17,18 showing the Graph structures of all the counties in the united states at three different months, and these graphs are induced dynamic graphs with other edge connections, dropped out random edges, with different active and inactive nodes. Each node in this graph visually represents the county, and the size of the node shows the

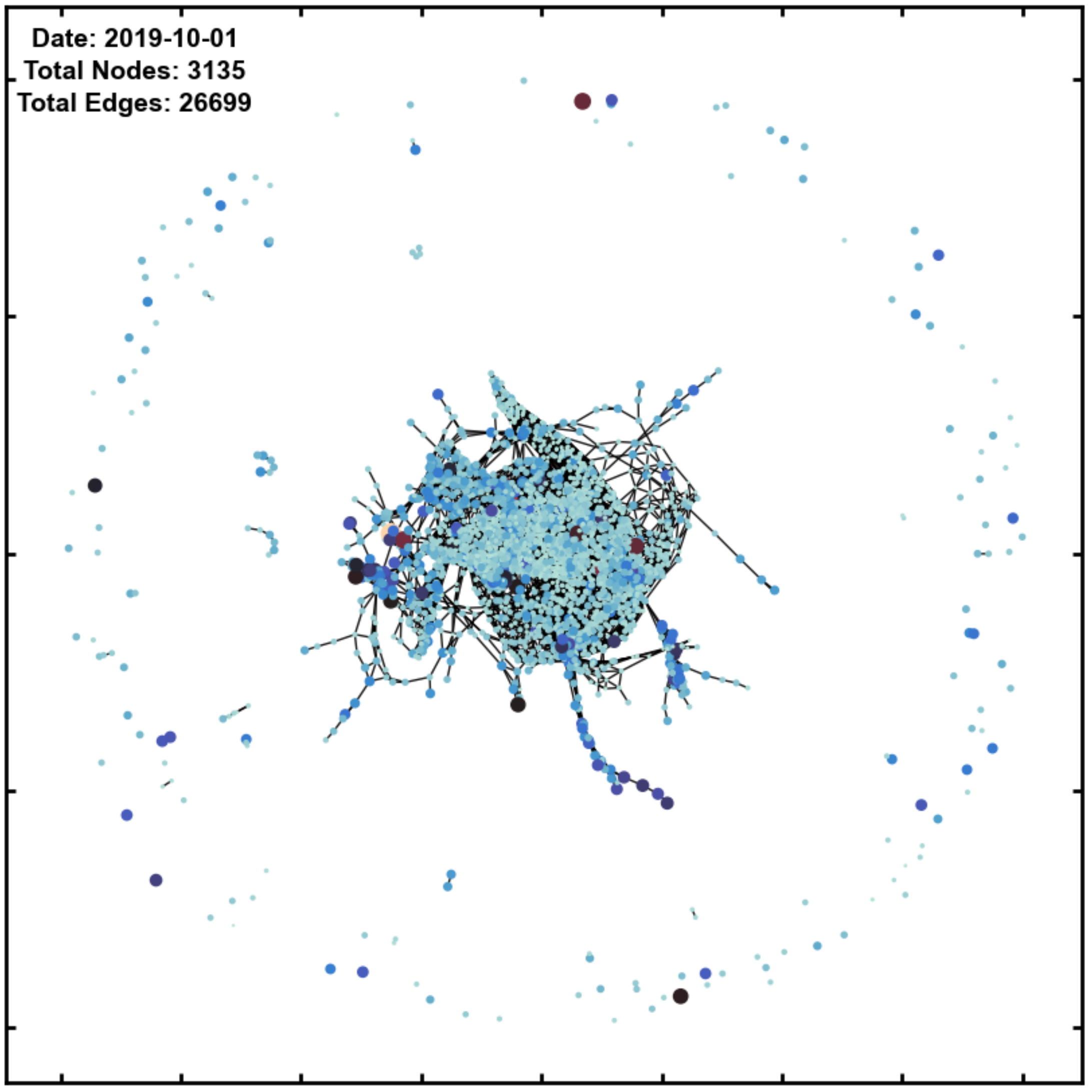


Figure 18: Data visualization of the graph structure on 2019-10-01

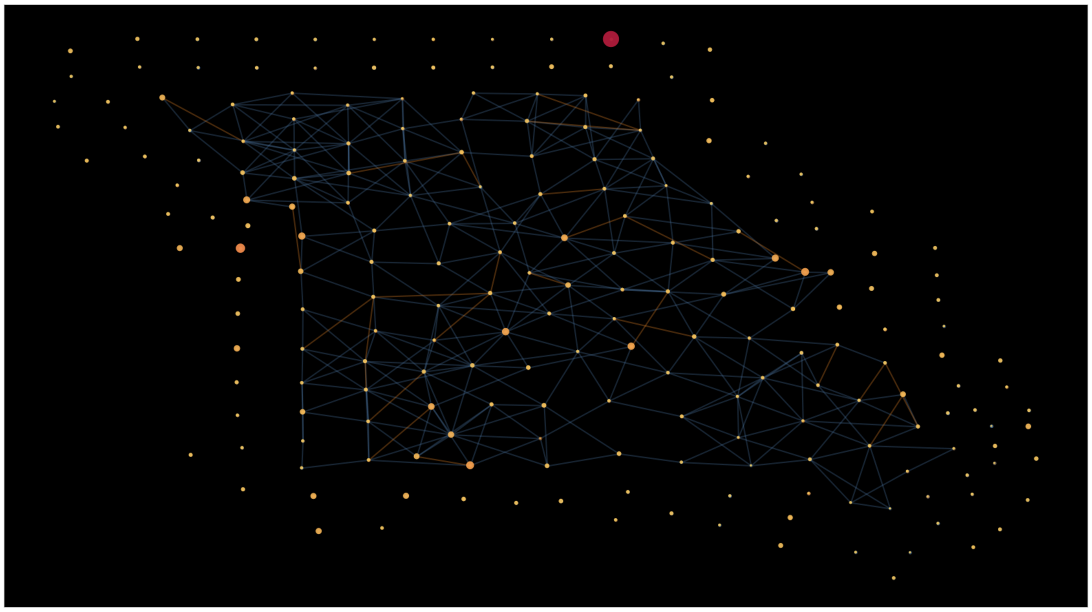
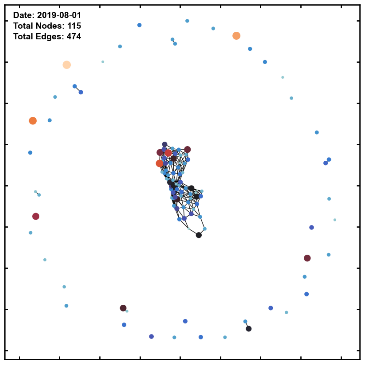


Figure 19: Network Graph of Missouri State with Distance Threshold of 50mile between county to county. Nodes are the counties, and edges are the distance between them.

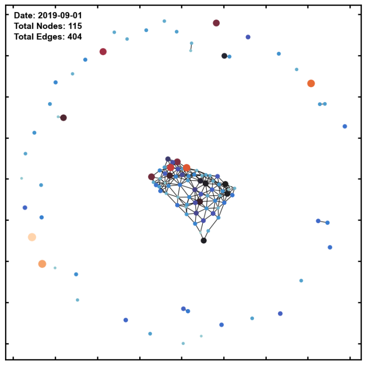
microbusiness density value at that county at that given time. The edge connection shows how far they are from each other, and all the connected nodes are the counties within a radius of 50 miles of other counties. The nodes outside the rim of a disconnected node are far-way nodes with no connections.

In this experiment, the area of interest is Missouri state-level counties and the temporal graphs within these counties; the reason for this is to compare the results with experiment 2 to validate the efficiency of the proposed framework. Figure 19 shows the geo-spatial distribution of Missouri state graph representation with counties as nodes and edges as the distance between each county.

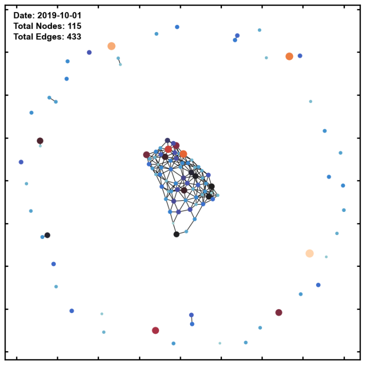
Figure 20 shows the county-level graphs' temporal snapshots of the first six months. Each node in this graph visually represents the county at the Missouri state level, counties including all the dots shown in Figure 20. The node size offers the microbusiness density value at that given time. Like the nationwide graphs, edge connection shows how far they are from each other, and all the connected nodes are the counties within a radius of 50 miles of other counties. One key observation of these graphs is that highly populated cities have higher business densities.



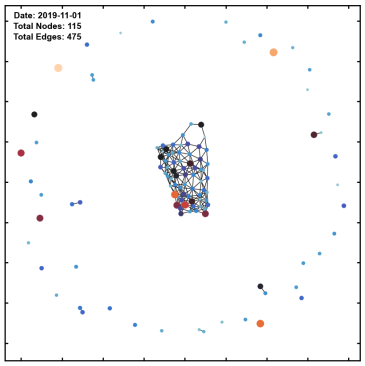
(a) Microbusiness Density of Missouri State as a graph 2019-08-01



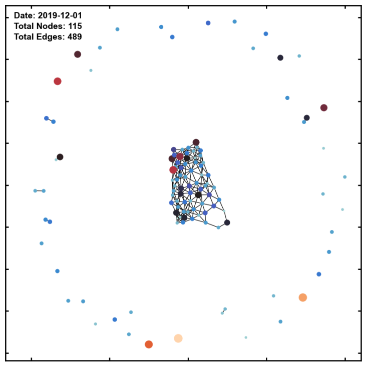
(b) Microbusiness Density of Missouri State as a graph 2019-09-01



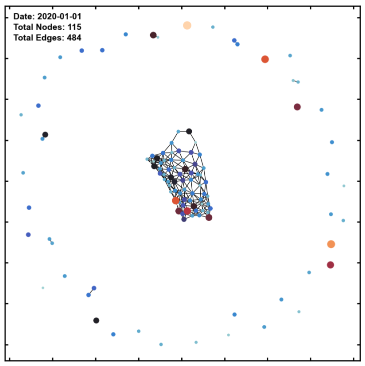
(c) Microbusiness Density of Missouri State as a graph 2019-10-01



(d) Microbusiness Density of Missouri State as a graph 2019-11-01



(e) Microbusiness Density of Missouri State as a graph 2019-12-01



(f) Microbusiness Density of Missouri State as a graph 2020-01-01

Figure 20: Comparison of Microbusiness Density of Missouri State for 6 Months, Each subplot displays the Graph representation of the data on a Graph illustrating the variation in the spatial distribution of micro businesses over time.

These experiments in this section are of different combinations with Attention-based Temporal Graph Convolutional Network architecture called A3TGCN [39], and these combinations are as follows:

1. **Static Graphs:** Experiments with static graph setting
    - Temporal Features
    - Temporal Features + Census Data
2. **Dynamic Graphs:** Experiments with dynamic graph setting
    - Temporal Features
    - Temporal Features + Census Data

4.4.2 Model Parameters

In this study, an Adam optimizer is used with an adaptive learning rate of 0.001 with early patience of 10 epochs. The data partition training dataset has 27 temporal graphs, the validation dataset have 3 temporal graphs, and the test dataset has 3 temporal graphs. The total epochs for each experiment are 250 epochs.

4.4.3 Static Graphs

This experiment used a static graph structure to forecast each Missouri county. Table 5 presents the results of the static graph experiments, displaying the data metrics of SMAPE, MAPE, and RMSE for the Missouri state-level forecasting. Additionally, Figure 21 demonstrates the three-month predictions vs. the ground truth values with temporal and census data node feature graph settings. Based on the metrics, when the node features are included with census data models, the SMAPE values are comparatively lesser than the other methods. These results suggest that incorporating additional data sources can help to improve the accuracy of time series forecasting, as demonstrated by the benefits of using census data in the model.

Table 5: Comparison of Missouri State Level Forecasting Metrics with Static Graph

Node Features	SMAPE	MAPE	RMSE
Temporal Features	0.26124	0.42934	0.0952
Temporal Features + Census Data	0.13147	0.29429	0.092

4.4.4 Dynamic Graphs

In these Experiments, a dynamic graph structure was utilized to forecast each Missouri county's time series. Table 6 displays the dynamic graph experiments results, showing the testing data metrics of SMAPE, MAPE, and RMSE for Missouri state-level forecasting. The SAMPE values of node features, including temporal features and census data, perform better than the others. Figure 22 illustrates the three-month predictions vs. the ground truth values with temporal data node feature graph settings. The dynamic graph can capture most of the trends of the counties and predict well overall.

Table 6: Comparison of Missouri State Level Forecasting Metrics with Dynamic Graph

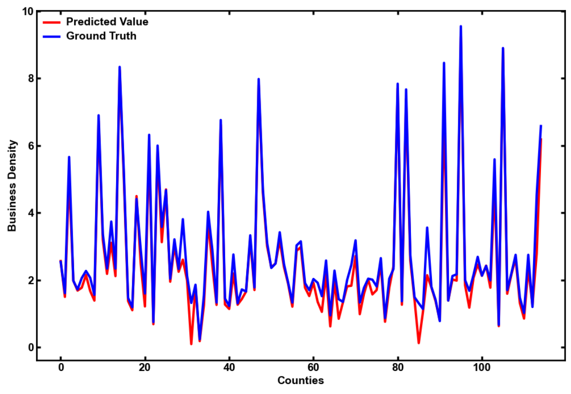
Node Features	SMAPE	MAPE	RMSE
Temporal Features	0.26163	0.43081	0.09452
Temporal Features + Census Data	0.13113	0.29401	0.09010

4.4.5 Comparisons between the Baselines and the Temporal Graph Networks

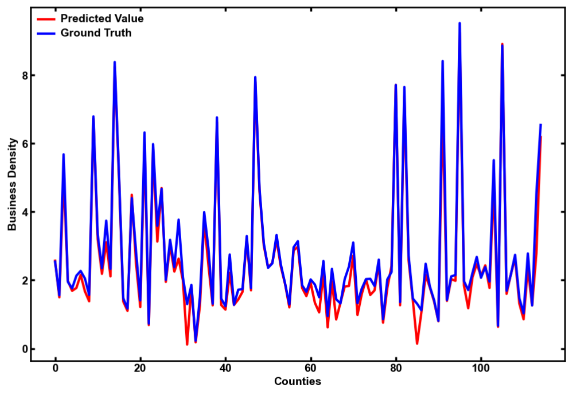
The dynamic graph neural network model, A3TGCN, outperforms all other methods in terms of all three evaluation metrics. Table 7 results suggest that the A3TGCN model can better capture the complex interdependencies and temporal dynamics between the county's time series data. The error metrics show a significant performance improvement. Thus, it can provide more accurate forecasts than traditional baseline and machine learning models used in Experiments 1 and 2.

Table 7: Comparison of Missouri State Level Forecasting of Temporal Graph and Baseline models

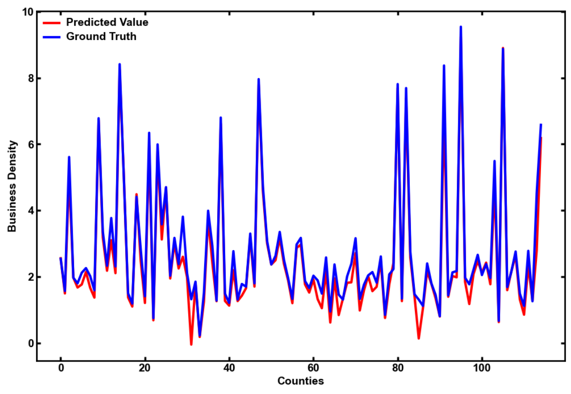
Forecast	RMSE	MAPE	SMAPE
A3TGCN(dynamic, all features)	0.09010	0.29401	0.13113
A3TGCN(static, all features)	0.092	0.29429	0.13147
holt_winters	0.10	2.59	2.66
last_naive	0.10	2.68	2.80
random_forest	0.14	4.33	4.28
season_naive	0.17	5.56	5.77



(a) Static Graph Predictions at 2022-08-01

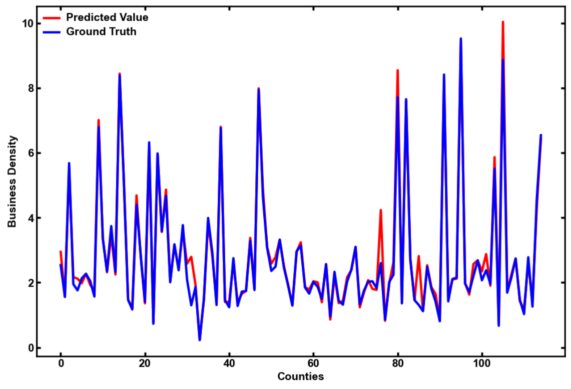


(b) Static Graph Predictions at 2022-09-01

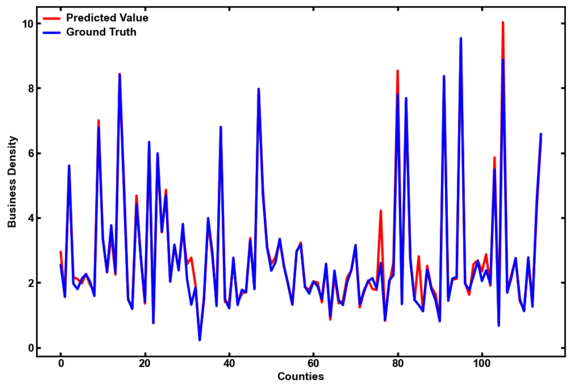


(c) Static Graph Predictions at 2022-10-01

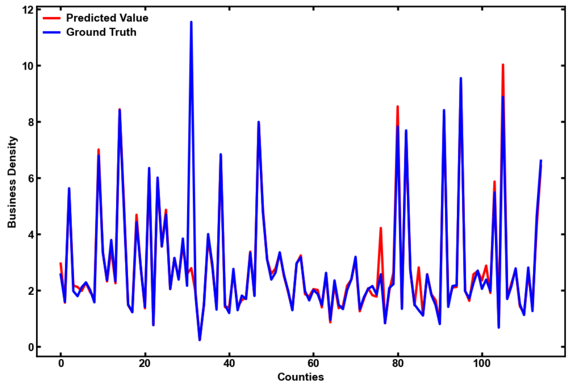
Figure 21: Temporal Graph Predictions with Static Graph for Missouri State Level Counties. Each subplot displays the microbusiness density predictions, illustrating the three-month variation.



(a) Dynamic Graph Predictions at 2022-08-01



(b) Dynamic Graph Predictions at 2022-09-01



(c) Dynamic Graph Predictions at 2022-10-01

Figure 22: Temporal Graph Predictions with Dynamic Graph for Missouri State Level Counties. Each subplot displays the microbusiness density predictions, illustrating the three-month variation.

CHAPTER 5

CONCLUSION AND FUTURE WORK

This thesis discusses various time series forecasting methods in chapter 2 of the related work. These are powerful yet simple mathematical and statistical models, where they have limitations. The interdependency capture within a time series network is impossible with existing statistical and machine learning models. Meanwhile, this application is growing daily in the current internet era. This application includes traffic predictions [39], stock predictions [35], disease spread patterns [47], etc. Recently, a new concept of deep learning emerged to take such network scenarios. One such idea is the temporal graph convolutional networks [39], which can capture both spatial information and material aspects of the given network. This is a new promising approach to several modern applications. So, a novel framework to tackle these limitations is needed.

5.1 Conclusion

In this thesis, we addressed the limitations of existing time series forecasting methods, which cannot capture interdependencies within a time series network. We discussed how the emergence of deep learning approaches, such as temporal graph convolutional networks, provides a promising solution to this problem by capturing both spatial and temporal dimensions of a given network. To validate the performance of the proposed attention-based temporal graph convolutional network framework, we conducted experiments on time series forecasting using traditional methods, machine learning models, and neural graph networks.

1. The first experiment assessed the baseline time series forecasting models with a single time series. We evaluated the performance of the Naive, Seasonal Naive, SARIMA, and Prophet models.
2. In the second experiment, we used the baseline models to forecast multiple time series. We added the sum of losses to compare their performance with the advanced temporal graph-based model. This experiment revealed that the Naive model was inefficient, while the Holt-Winters smoothing method outperformed the other baseline models.
3. Finally, we used a temporal graph neural network model called A3TGCN to perform time series forecasting and compare its results with the traditional methods and machine learning models for all the counties in Missouri State. Our results showed that the A3TGCN model outperformed all the other models regarding MAPE, SMAPE, and root mean square error (RMSE) of individual time series predictions.

In conclusion, this thesis proposed a new framework for implementing and validating attention-based temporal graph convolutional networks for time series forecasting. Our experiments demonstrate that the A3TGCN model can effectively capture the interdependencies within a time series network and provide more accurate predictions than traditional methods and machine learning models. This study contributes to the growing field of deep learning approaches for time series forecasting and provides a promising direction for future research.

5.2 Future Work

In terms of future work, this study opens up several research directions.

5.2.1 Bottleneck Issues and A3TGCN Complexity Analysis

- Study of Under-Reaching and Over-Squashing as challenging Graph Networks bottlenecks [48], which also apply to the A3TGCN model.
- Under-reaching refers to the issue where certain nodes in the graph fail to receive sufficient information from their neighboring nodes, leading to an incomplete understanding of the traffic patterns.
- Over-squashing occurs when nodes overly compress or summarize the received information, losing important details.
- A thorough analysis of these bottlenecks is crucial for fine-tuning the message parsing ability of the A3TGCN model. This analysis could investigate different architectures, activation functions, or attention mechanisms to overcome these challenges and improve the model's performance.

5.2.2 Dynamic Graph Conversion Experimentations

- In the current study, a random drop of edges was implemented to reduce the graph's complexity. However, exploring alternative methods, such as weighted drop and adding edges, could be beneficial.
- Weighted drop and addition of edges would involve assigning different weights to the edges based on their importance or relevance to the traffic patterns. This approach could improve prediction accuracy by selectively retaining or introducing edges that carry important information for the model's predictions.

5.2.3 Level of Detail Studies in Model Training

- The A3TGCN model offers the ability to choose the level of detail in the graph based on physical boundaries. This means that different regions or areas within the overall network can be represented with varying levels of granularity.
- Conducting studies to determine the optimal level of detail for model training is essential. It would involve evaluating the impact of different granularity on the model's accuracy, computational efficiency, and generalizability.
- Finding the right level of detail will ensure that the model captures the basic patterns while maintaining a manageable computational complexity, leading to more accurate predictions.

By addressing these future areas, this work can further refine and improve the A3TGCN model's predictive capabilities. The in-depth analysis of bottleneck issues, exploration of dynamic graph conversion techniques, and understanding of the impact of the level of detail in model training will contribute to advancing the field of spatiotemporal network prediction and enable more accurate and efficient variable forecasting. Exploring the application of A3TGCN for time series forecasting in other areas, such as finance, healthcare, and transportation, can lead to new insights and improvements in forecasting accuracy. Investigating the impact of incorporating additional data sources, such as social media and weather data, can provide a more comprehensive understanding of the factors influencing time series behavior. Exploring the interpretability of the proposed model can help understand the underlying factors and patterns affecting time series forecasting and provide more transparent decision-making in real-world applications. Lastly, exploring the scalability of the proposed model for more extensive networks can be an exciting area of research.

REFERENCE LIST

[1] Mitchell, "Machine learning," Wikimedia Commons, p. 87, 1997. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/f/ff/Rosenblattperceptron.png?20121024111529

[2] A. McAfee, E. Brynjolfsson, T. H. Davenport, D. Patil, and D. Barton, "Big data: the management revolution," *Harvard Business Review*, vol. 90, no. 10, pp. 60–68, 2012.

[3] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, A. Hung Byers *et al.*, *Big Data: The Next Frontier for Innovation, Competition, and Productivity*. McKinsey Global Institute, 2011.

[4] J. Gantz, D. Reinsel *et al.*, "Extracting value from chaos," *IDC iview*, vol. 1142, no. 2011, pp. 1–12, 2011.

[5] M. Hilbert and P. López, "The worldâs technological capacity to store, communicate, and compute information," *Science*, vol. 332, no. 6025, pp. 60–65, 2011.

[6] J. G. De Gooijer and R. J. Hyndman, "25 years of time series forecasting," *International Journal of Forecasting*, vol. 22, no. 3, pp. 443–473, 2006.

[7] L. Li, B. A. Prakash, and C. Faloutsos, "Parsimonious linear fingerprinting for time series," Carnegie Mellon University Pittsburgh PA School of Computer Science, Tech. Rep., 2010.

[8] E. S. Gardner Jr, “Exponential smoothing: The state of the art,” *Journal of Forecasting*, vol. 4, no. 1, pp. 1–28, 1985.

[9] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2015.

[10] M. Qi and G. P. Zhang, “Trend time–series modeling and forecasting with neural networks,” *IEEE Transactions on Neural Networks*, vol. 19, no. 5, pp. 808–816, 2008.

[11] R. Adhikari and R. K. Agrawal, “An introductory study on time series modeling and forecasting,” *ArXiv Preprint arXiv:1302.6613*, 2013.

[12] B. Yu, H. Yin, and Z. Zhu, “Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting,” *ArXiv Preprint arXiv:1709.04875*, 2017.

[13] M. S. Ahmed and A. R. Cook, *Analysis of Freeway Traffic Time-series Data by Using Box-Jenkins Techniques*, 1979, no. 722.

[14] B. M. Williams and L. A. Hoel, “Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results,” *Journal of Transportation Engineering*, vol. 129, no. 6, pp. 664–672, 2003.

[15] J. Zhang, Y. Yao, and Z. X. Feng, “Ming,” *Development and Evaluation of a Patient-reported Outcome (PRO) Scale for Breast Cancer*, vol. 16, no. 18, pp. 8573–8578, 2015.

[16] G. E. Box, “Gm jenkins time series analysis: Forecasting and control,” *San Francisco, Holdan-Day*, 1970.

[17] S. Makridakis and M. Hibon, “The m3-competition: Results, conclusions and implications,” *International Journal of Forecasting*, vol. 16, no. 4, pp. 451–476, 2000.

[18] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, “Statistical and machine learning forecasting methods: Concerns and ways forward,” *PloS one*, vol. 13, no. 3, p. e0194889, 2018.

[19] S. Makridakis and E. a. Spiliotis, “The m4 competition: Results, findings, conclusion and way forward,” *International Journal of Forecasting*, vol. 34, no. 4, pp. 802–808, 2018.

[20] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, “The accuracy of machine learning (ml) forecasting methods versus statistical ones: Extending the results of the m3-competition,” in *Working Paper, University of Nicosia*. Institute for the Future, 2017.

[21] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*. OTexts, 2018.

[22] M. Xie, C. Sandels, K. Zhu, and L. Nordström, “A seasonal arima model with exogenous variables for elspot electricity prices in sweden,” in *2013 10th International Conference on the European Energy Market (EEM)*. IEEE, 2013, pp. 1–4.

[23] C. Wu, J. Wang, and Y. Hao, “Deterministic and uncertainty crude oil price forecasting based on outlier detection and modified multi-objective optimization algorithm,” *Resources Policy*, vol. 77, p. 102780, 2022.

[24] E. D. Spyrou, I. Tsoulos, and C. Stylios, “Applying and comparing lstm and arima to predict co levels for a time-series measurements in a port area,” *Signals*, vol. 3, no. 2, pp. 235–248, 2022.

[25] R. G. Brown, “Statistical forecasting for inventory control,” 1959.

[26] C. C. Holt, “Forecasting seasonals and trends by exponentially weighted moving averages,” *International journal of forecasting*, vol. 20, no. 1, pp. 5–10, 2004.

[27] P. R. Winters, “Forecasting sales by exponentially weighted moving averages,” *Management science*, vol. 6, no. 3, pp. 324–342, 1960.

[28] T. Mureş and N. Iorga, “Neural networks versus box-jenkins method for turnover forecasting: A case study on the romanian organisation,” *Transformations in Business & Economics*, vol. 16, no. 1, p. 40, 2017.

[29] S. A. Hamid and A. Habib, “Financial forecasting with neural networks,” *Academy of Accounting and Financial Studies Journal*, vol. 18, no. 4, p. 37, 2014.

[30] A. Bredahl Kock and T. Teräsvirta, “Forecasting macroeconomic variables using neural network models and three automated model selection techniques,” *Econometric Reviews*, vol. 35, no. 8-10, pp. 1753–1779, 2016.

[31] J. Wang and J. Wang, "Forecasting stochastic neural network based on financial empirical mode decomposition," *Neural Networks*, vol. 90, pp. 8–20, 2017.

[32] G. Ristanoski, W. Liu, and J. Bailey, "Time series forecasting using distribution enhanced linear regression," in *Advances in Knowledge Discovery and Data Mining: 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14-17, 2013, Proceedings, Part I 17*. Springer, 2013, pp. 484–495.

[33] C.-J. Lu, T.-S. Lee, and C.-C. Chiu, "Financial time series forecasting using independent component analysis and support vector regression," *Decision Support Systems*, vol. 47, no. 2, pp. 115–125, 2009.

[34] N. I. Sapankevych and R. Sankar, "Time series prediction using support vector machines: A survey," *IEEE Computational Intelligence Magazine*, vol. 4, no. 2, pp. 24–38, 2009.

[35] L. Khaidem, S. Saha, and S. R. Dey, "Predicting the direction of stock market prices using random forest," *ArXiv Preprint arXiv:1605.00003*, 2016.

[36] H. Tyralis and G. Papacharalampous, "Variable selection in time series forecasting using random forests," *Algorithms*, vol. 10, no. 4, p. 114, 2017.

[37] C. L. Giles, S. Lawrence, and A. C. Tsoi, "Noisy time series prediction using recurrent neural networks and grammatical inference," *Machine Learning*, vol. 44, no. 1-2, p. 161, 2001.

[38] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[39] J. Zhu, Y. Song, L. Zhao, and H. Li, “A3t-gcn: Attention temporal graph convolutional network for traffic forecasting,” 2020.

[40] G. Jin, Y. Liang, Y. Fang, J. Huang, J. Zhang, and Y. Zheng, “Spatio-temporal graph neural networks for predictive learning in urban computing: A survey,” 2023.

[41] Y. Li, F. Zhao, Z. Chen, Y. Fu, and L. Ma, “Multi-behavior enhanced heterogeneous graph convolutional networks recommendation algorithm based on feature-interaction,” *Applied Artificial Intelligence*, vol. 37, no. 1, p. 2201144, 2023. [Online]. Available: https://doi.org/10.1080/08839514.2023.2201144

[42] H. Zhou, D. Ren, H. Xia, M. Fan, X. Yang, and H. Huang, “Ast-gnn: An attention-based spatio-temporal graph neural network for interaction-aware pedestrian trajectory prediction,” *Neurocomputing*, vol. 445, pp. 298–308, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092523122100388X

[43] Godaddy.com, “Microbusiness density data hub,” Available at: https://www.godaddy.com/ventureforward/microbusiness-datahub/, 2022, accessed Day Month Year.

[44] B. Chennoju, “Microbusiness density dynamic graph network visualization,” 2023. [Online]. Available: https://public.tableau.com/app/profile/bhuvan.kumar.chennoju/viz/network_plot/Dashboard1

[45] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, “T-GCN: A temporal graph convolutional network for traffic prediction,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3848–3858, sep 2020. [Online]. Available: https://doi.org/10.1109\%2Ftits.2019.2935152

[46] B. Yu, H. Yin, and Z. Zhu, “Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting,” in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, jul 2018. [Online]. Available: https://doi.org/10.24963\%2Fijcai.2018\%2F505

[47] G. Panagopoulos, G. Nikolentzos, and M. Vazirgiannis, “Transfer graph neural networks for pandemic forecasting,” 2021.

[48] U. Alon and E. Yahav, “On the bottleneck of graph neural networks and its practical implications,” 2021.

VITA

Bhuvan Kumar Chennoju earned his Bachelor of Technology in Mechanical Engineering in 2018 from the National Institute of Technology in Hamirpur, India. Afterward, he worked as an engineer and research assistant, focusing on manufacturing operations research, fundamental combustion research, and data analysis. In 2021, he joined the Computer Science master's program at the School of Computing and Engineering, University of Missouri-Kansas City. Bhuvan participated in a hackathon centered around Augmented Reality and Artificial Intelligence in 2022 and a research-a-thon for temporal predictive models using GNN in 2023. He also contributed to the NSF Smart Connected Community project, collecting neighborhood data and visualizing the information gathered from mobile applications. In 2022, Bhuvan interned as a Software Developer at T-Mobile USA. Additionally, he worked as a graduate teaching assistant for several courses, including Introduction to statistical learning and Deep Learning. For his MS thesis project, he is developing a temporal graph network-based model for predicting finance business density across various counties and has shared multiple projects on GitHub.