FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Application of Graph Neural Networks in Road Traffic Forecasting for Intelligent Transportation Systems

Ana Clara Moreira Gadelho

Mestrado em Engenharia Informática e Computação

Supervisor: Dr. Daniel Augusto Gama de Castro Silva

Second Supervisor: Dr. Rosaldo José Fernandes Rossetti

July 21, 2023

Application of Graph Neural Networks in Road Traffic Forecasting for Intelligent Transportation Systems

Ana Clara Moreira Gadelho

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Prof. Rui Camacho
External Examiner: Prof. Marco Veloso
Supervisor: Prof. Daniel Castro Silva

July 21, 2023

This work is a result of project DynamiCITY: Fostering Dynamic Adaptation of Smart Cities to Cope with Crises and Disruptions, with reference NORTE-01-0145-FEDER-000073, supported by Norte Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF).

Cofinanciado por:

Resumo

A previsão de tráfego é um aspecto crucial dos Sistemas de Transporte Inteligentes, uma vez que tem o potencial de melhorar a mobilidade e eficiência do transporte nas cidades, reduzindo os custos e minimizando o impacto ambiental. A tarefa de previsão de tráfego é um desafio complexo, uma vez que envolve lidar com a rápida evolução e o dinamismo do tráfego, que é afectado por vários factores, tais como acidentes, cortes de estradas, eventos sociais e meteorológicos. Para além disso, o fluxo de tráfego é caracterizado por dependências espaciais e temporais, em que o estado do tráfego num determinado local é influenciado por o tráfego noutros locais e o estado do tráfego num determinado momento depende do tráfego passado.

Nos últimos anos, as *Graph Neural Networks* (GNNs) têm ganho cada vez mais atenção no terreno de *Deep Learning*, demonstrando desempenho ao nível de estado da arte em várias aplicações. As GNNs são particularmente adequadas a problemas de previsão de tráfego, uma vez que têm a capacidade de capturar tanto dependências espaciais como temporais nos dados.

Esta dissertação explora a utilização de GNNs na previsão de tráfego rodoviário. O estudo fornece uma revisão da literatura existente sobre técnicas de previsão de tráfego e investiga o potencial das GNNs no tratamento das relações complexas entre as condições de tráfego, tanto no tempo como no espaço. A investigação tem como objetivo avaliar o desempenho das GNNs neste problema e complementar esse estudo com a investigação do impacto da utilização de técnicas de imputação de dados em falta e dos factores externos nos resultados da previsão. O estudo também tem como objetivo avaliar a generalização dos modelos GNN em diferentes datasets, incluindo datasets de *benchmarking* e o dataset de caso de uso do projeto DynamiCITY, que esta dissertação integra.

A avaliação empírica dos modelos demonstra a sua eficácia no tratamento das relações complexas entre as condições de tráfego no tempo e no espaço. Os resultados da investigação mostram que as GNNs superam vários modelos baseados em séries temporais comummente utilizados na literatura, em linha com outros trabalhos sobre este tema. As técnicas de imputação de dados em falta também apresentaram melhores resultados, especialmente nos conjuntos de dados com mais dados em falta. No entanto, a utilização de dados meteorológicos não melhorou os resultados obtidos.

Em termos gerais, esta dissertação contribui para a crescente investigação sobre a utilização de GNNs na previsão de tráfego e demonstra o seu potencial para melhorar os sistemas de transporte.

Abstract

Traffic forecasting is a crucial aspect of Intelligent Transportation Systems, as it has the potential to improve the mobility and efficiency of transportation in cities while reducing costs and minimizing environmental impact. The task of traffic forecasting is a challenging one, as it involves predicting the rapidly changing and dynamic nature of traffic, which is affected by various factors such as accidents, road closures, social events, and weather. Additionally, traffic flow is characterized by both spatial and temporal dependencies, where the state of traffic in one location is influenced by traffic in other locations and the state of traffic at a particular time is dependent on past traffic patterns.

In recent years, Graph Neural Networks (GNNs) have gained increasing attention in the field of deep learning, demonstrating state-of-the-art performance in various applications. GNNs are particularly well suited for traffic forecasting problems, as they have the ability to capture both spatial and temporal dependencies in the data.

This dissertation explores the application of GNNs in road traffic forecasting. The study provides a review of existing literature on traffic forecasting techniques and investigates the potential of GNNs in handling the complex relationships between traffic conditions in time and space. The research aims to evaluate the performance of GNNs in this problem and complement that study by investigating the impact of using missing data imputation techniques and external factors on the prediction results. The study also aims to evaluate the generalizability of the GNN models across different datasets, including benchmarking datasets and DynamiCITY's use-case dataset.

The empirical evaluation of the models demonstrates their effectiveness in handling the complex relationships between traffic conditions in time and space. The results of the research show that GNNs outperform several commonly used time-series-based models in the literature, in line with other works on the topic. The missing data imputation techniques also showed improved results, especially on the datasets with more missing data. However, the use of weather data did improve the results obtained.

Overall, this dissertation contributes to the growing body of research on the use of GNNs in traffic forecasting and demonstrates their potential to improve transportation systems.

Keywords: traffic forecasting, intelligent transportation systems, graph neural networks, spatio-temporal dependency, deep learning, missing data, external factors

Acknowledgments

First, I would like to thank Professor Daniel Silva and Professor Rosaldo Rossetti, supervisors of this dissertation, for their guidance and encouragement throughout the duration of this dissertation process. Without their support in the most stressful phases this work could not be the way it is. I would like to acknowledge the support and resources provided by LIACC and the DynamiCITY research project. The access to computing facilities and data provided were crucial in conducting the experiments and analysis for this research.

I also want to express my gratitude for my family, in particular my parents and brother, for always believing in me and giving me encouragement for all the challenges I face.

Finally, I want to thank all my friends, that accompanied me throughout this five years and in particular this last few months . Without them this journey would not have been the same.

Clara Gadelho

Contents

List of Figures

List of Tables

Listings

Acronyms and Abbreviations

AI	Artificial Intelligence
Agg	Aggregation
Att	Attention
ARIMA	Autoregressive Integrated Moving Average
ASTGCN	Attention Based Spatial-Temporal Graph Convolutional Network
ChebConv	Chebyshev Spectral Graph Convolution
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
DSTGCN	Dynamic Spatial-Temporal Graph Convolutional Network
FNN	Feedforward Neural Network
GAT	Graph Attention Network
GCN	Graph Convolution Network
GMAN	Graph Multi-Attention Network
GNN	Graph Neural Network
GRU	Gated Recurrent Unit
GTCN	Temporal Graph Convolutional Network
ITS	Intelligent Transportation Systems
LOCF	Last Observation Carried Forward
LSTM	Long Short-term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MICE	Multivariate Imputation by Chained Equations
ML	Machine Learning
MLP	Multilayer Perceptron
NN	Neural Network
NOCB	Next Observation Carried Backwards
N/A	Non Applicable
PeMS	Caltrans Performance Measure Systems
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SLC	Structure Learning Convolution
STGCN	Spatio-Temporal Graph Convolutional Network
ST-UNet	Spatio-Temporal U-Network for Graph-structured Time Series Modeling
SVM	Support Vector Machine
T-GCN	Temporal Graph Convolutional Network
VAR	Vector Auto Regressive Model
VCI	*Via de Cintura Interna* (Porto's Ring Road)
WHO	World Health Organization

Chapter 1

Introduction

1.1 Context

With the acceleration of urbanization and the rapid growth of urban population, great pressure is being placed on urban traffic management (United Nations, 2017). Expanding cities face numerous challenges related to transportation, including increased air pollution and worsening traffic congestion (W. Jiang & Luo, 2022). According to the World Health Organization (WHO, 2022), the transportation sector leads to a significant portion of air pollution, with more than seven million people dying each year due to this crisis. With this in mind, great effort is nowadays being placed into developing Intelligent Transportation Systems (ITS), which could significantly improve the lives of residents in future cities (Shahid, Shah, Khan, Maple, & Jeon, 2021). In the context of ITS, Traffic forecasting is seen as an essential step in improving the efficiency of the transportation system and alleviating transportation-related problems (J. Lu, Li, Li, & Al-Barakani, 2021). This is a complex task since traffic is very dynamic and involves dependencies in terms of space and time (W. Jiang & Luo, 2022). So, over the years, researchers have been iterating over different ways of improving traffic predictions. This dissertation is being prepared within the scope of an academic research project: DynamiCITY. DynamiCITY aims to create a virtual environment for the design and implementation of solutions for mobility within smart cities. It consists of an open-data platform and a multi-agent-based decision-support system that allows researchers and practitioners to explore the relationship between the city and its transport system, focusing on climate neutrality, safety, and accessibility. The DynamiCITY is being developed as a collaboration between three research laboratories of FEUP: LIACC, CITTA, and SYSTEC. It will represent an important R&D infrastructure available to the whole research community.

1.2 Aim and Goals

The main aim of this dissertation is to investigate the use of machine learning models for road traffic forecasting and to evaluate their performance in real-world scenarios. To achieve this, state-of-the-art approaches to traffic forecasting tasks will be explored, in order to find the current trend

of research and explore how to properly deal the main challenges of this problem, such as the complex relationships between traffic conditions in time, space and external factors. This research will then be put to use in developing and implementing a machine learning model based on the current trend of research for road traffic forecasting, considering factors such as weather and time of day that may affect traffic patterns. Being part of a research project that focuses on smart cities, the ultimate goal of this dissertation is to be able to use this model to make accurate and reliable traffic predictions to be incorporated into Intelligent Transportation Systems across different road networks.

1.3 Contributions

The contributions of this dissertation can be summarized as follows:

- A comprehensive review of the existing literature of traffic forecasting, with a special focus on techniques based in Graph Neural Networks(GNN).
- Comparison of the performance of GNNs with other commonly used traffic prediction models in the literature.
- Investigation of the impact of using missing data imputation techniques on traffic prediction performance of Graph Neural Networks (GNNs).
- Examination of the effectiveness of incorporating external data, such as weather conditions, in improving the accuracy of traffic predictions.
- Evaluation of the generalizability of the GNN models across different datasets, including benchmarking datasets and DynamiCITY's use-case dataset.
- Development of a GNN-based traffic prediction framework that can be applied to different traffic networks.

Overall, the results of this research contributed to a better understanding of the potential of GNNs in the context of traffic forecasting.

1.4 Outline

The dissertation begins with an introduction that sets the stage for the research. It provides the necessary context for understanding the study and outlines the aims and goals that the research seeks to achieve. The expected contributions of the work are also highlighted, giving an overview of what can be expected from the document.

The background chapter, Chapter 2, delves into the main areas of study. First, exploring the concept of Intelligent Transportation Systems (ITS) and their relevance to the research topic. Secondly, the chapter focuses on the traffic forecasting problem itself. It discusses various traffic

prediction techniques and categorizes the different types of traffic prediction problems. Moreover, the chapter introduces Graph Neural Networks (GNNs) as a potential solution to the traffic forecasting problem. The architecture of GNNs is explained, with a particular emphasis on the attention mechanism.

Chapter 3 reviews the existing literature and research in the field of traffic forecasting. It provides an overview of various techniques and approaches that have been employed, including statistical methods, ensemble learning, hybrid methods, machine learning, and deep learning techniques. Special attention is given to the application of Graph Neural Networks in traffic forecasting. Furthermore, benchmarking datasets used in the field are discussed, and a gap analysis is presented to identify areas where the current research can make valuable contributions.

The methodological approach chapter, Chapter 4, outlines the methodology followed in the research. It begins by clearly stating the problem and formalizing it. The chapter then presents the implementation pipeline, detailing the steps involved in the research process. Validation is addressed, including the metrics used to evaluate the models and the baselines against which the models are compared. Additionally, the chapter includes a risk analysis, considering potential challenges and limitations associated with the research.

Moving on to the implementation in Chapter 5, the focus is on the practical aspects of the research. It discusses the technologies used in the research and describes the data retrieval and pre-processing steps. This includes the handling of traffic data, weather data, and event data. The chapter also explains the process of generating the graph dataset, covering aspects such as adjacency matrix generation, traffic speed normalization, and the incorporation of additional features. Then the base Graph Neural Network architecture is described, including the different layers and network hyperparameters. Moreover, techniques for handling missing data and incorporating external data, such as weather and event data, are explored.

In the empirical evaluation, Chapter 6, the research findings are presented and analyzed. The performance and comparisons of the base Graph Neural Network architecture on each dataset are discussed. The effect of different hyperparameters is evaluated, and the impact of handling missing data and incorporating weather data is explored. Global comparisons are made to gain insights into the overall performance of the models. The results are also discussed, providing an understanding of the empirical findings.

Finally, the dissertation concludes with Chapter 7. The main findings and contributions of the research are highlighted, and the limitations and potential areas for future research are addressed. This serves as a reflection on the research journey and offers closure to the document.

Chapter 2

Background

This introductory chapter presents a summary of the research themes explored in this dissertation so the reader can become familiarized with the fundamental concepts necessary to comprehend it. Accordingly, this chapter provides an overview of the main topics covered in this dissertation: ITS, Traffic Prediction and Graph Neural Networks.

2.1 Intelligent Transportation Systems

Intelligent Transportation Systems (ITS) are advanced technologies and systems that are used to improve the safety, efficiency, and effectiveness of transportation systems (Irawan, Yusuf, & Prihatmanto, 2020). These systems use various technologies, such as computer systems, wireless communication, and sensors, to gather and analyze transportation data, and then use this data to control and optimize the performance of the transportation system at hand (Ang, Seng, Ngharamike, & Ijemaru, 2022). ITS systems can be applied to a wide range of transportation modes, including highways, public transit, and active transportation (such as biking and walking). Examples of ITS include traffic management systems, advanced traveller information systems, and automated vehicle systems. The goal of ITS is to make transportation safer, more efficient, and more reliable, while also reducing congestion, emissions, and other negative impacts of transportation on the environment.

2.2 The Traffic Forecasting Problem

Traffic forecasting is the process of predicting future traffic patterns and conditions. This can include predicting traffic volumes, travel times, and congestion levels on roads, highways, and other transportation networks. Traffic forecasting is a critical component of transportation planning and management, as it enables decision-makers to proactively address potential congestion, improve safety, and allocate resources effectively. There are various methods used for traffic forecasting, including traditional statistical models and machine learning (Y. Zhao, Zhang, An, & Liu, 2018). The goal of traffic forecasting is to provide accurate and reliable predictions, which can be used to

inform a range of decisions, from planning and designing new transportation systems to optimizing existing networks.

2.2.1 Traffic Prediction

In the context of artificial intelligence and machine learning, the tasks that try to solve traffic forecasting problems are referred to as traffic predictions. The terms forecast and prediction can sometimes be used interchangeably. Still, for the sake of rigour, when referring to the problem in the transportation domain, the term should be traffic forecasting, whilst the machine learning approaches should be referred to as traffic prediction techniques.

Road traffic prediction plays a crucial role in traffic management and ITS, and it can be used to optimize traffic flow, reduce congestion, improve safety, and provide better route guidance and estimated time of arrival for travellers. With the current advance in technology and the availability of real-time traffic data, it's becoming more accurate, reliable and can be used for different purposes (Alam, Farid, & Rossetti, 2019).

Traffic prediction is a complex task that poses several challenges. One of the main challenges is obtaining high-quality data from various sources such as traffic sensors, GPS, and social media, which can be difficult to obtain, clean and integrate (Kołodziej, Hopmann, Coppa, Grzonka, & Widłak, 2022). Traffic patterns are influenced by various factors, such as weather, special events, and human behaviour, making it difficult to model and predict traffic conditions. Handling spatio-temporal dependencies is also a challenge as traffic conditions are influenced by the interactions between different parts of the road network, and these interactions can vary over time. With the increasing amount of data being generated by multiple sources, scalability becomes a major challenge (Laña, Del Ser, Velez, & Vlahogianni, 2018). Real-time prediction is also challenging as it requires predictions to be made quickly and accurately. Choosing the right model and evaluating its performance can also be difficult due to the lack of ground truth data and the dependence of results on the specific application and data used.

2.2.2 Categorization of Traffic Prediction Problems

Traffic prediction is a broad topic that is categorized into different sub-problems in the literature, considering the different dimensions of this problem.

- **Data Source:** Regarding the data source for the problem, it's possible to have them categorized into moving, interval and point sources (Angarita-Zapata, Masegosa, & Triguero, 2019) (W. Jiang & Luo, 2022). Interval data sources, such as Automatic Toll Collection systems, Video cameras, and License Plate Recognition, differ from point detectors in that they offer a more comprehensive understanding of vehicle movement along a particular segment of the road. These sources are equipped with sensors placed at two fixed points, which calculate measures such as travel time between the points (Angarita-Zapata et al., 2019) (Mori, Mendiburu, Álvarez, & Lozano, 2015). Moving data sources is the newest since

it appeared with the use of Global Positioning Systems (GPSs) in vehicles and other devices. These can provide individual traffic data related to vehicles' trajectories on the roads and more detailed traffic information (Angarita-Zapata et al., 2019) (Castro, Zhang, & Li, 2012). Point data sources are placed at specific locations on the roads to detect the presence of nearby vehicles. These sensors, called point data sources, provide information such as traffic flow (number of vehicles passing per time unit), occupancy (the percentage of time that a sensor is detecting a vehicle), and density (the number of vehicles per unit length of the road). Common types of sensors used in this category include loop detectors, microwave radars, laser radar sensors, infrared sensors, among others (Lopes, Bento, Huang, Antoniou, & Ben-Akiva, 2010) (Mori et al., 2015). This is the data source that will be used for this dissertation.

- **Context:** The context provides information on transportation environments wherein traffic predictions can occur and can be defined as urban and freeway (Angarita-Zapata et al., 2019) (E. Vlahogianni, Golias, & Karlaftis, 2004). The context is important to define the particularities of the traffic in cause. In urban settings, the traffic speed is lower, and there are more road intersections and obstacles, wherein highway settings we can expect a higher speed and fewer road intersections. This dissertation focuses on highway problems.

- **Prediction Horizon:** The predictions can be short-term, which are usually defined as predictions for a time interval smaller than 60 minutes or long-term if the prediction window is bigger than 60 minutes (Irawan et al., 2020) (K. Lee & Rhee, 2022) (Angarita-Zapata et al., 2019). This work focuses on short-term prediction.

- **Prediction Output:** Traffic prediction tasks can be classified based on the traffic state to be predicted. Predictions related to traffic flow, speed, and demand are the most widely addressed and are typically handled as distinct problems, while other types are generally grouped together(W. Jiang & Luo, 2022). This work focuses on traffic speed prediction.

- **Traffic Representation:** The prediction can also be defined in terms of how traffic states are defined. These are graph/network-based representations or grid-based representations. Graph-based (R. Jiang et al., 2021). Grid-based traffic prediction refers to the use of a grid-based data structure, where the area of interest is divided into a regular grid of cells, and traffic data is collected and analyzed at each cell. This approach assumes that traffic conditions are homogeneous within each cell and that the data collected at each cell can be used to make predictions about traffic conditions in that cell (D. Chen et al., 2021) (Schörner, Hubschneider, Härtl, Polley, & Zöllner, 2019) (X. Zhou, Shen, & Huang, 2019). On the other hand, Graph-based prediction refers to the use of a graph-based data structure, where the area of interest is represented as a network of interconnected nodes and edges. This approach is based on the assumption that traffic conditions are influenced by the relationships and interactions between different parts of the network (Cui, Henrickson, Ke, & Wang, 2018) (L. Zhao et al., 2020). This work solely focuses on graph-based representations.

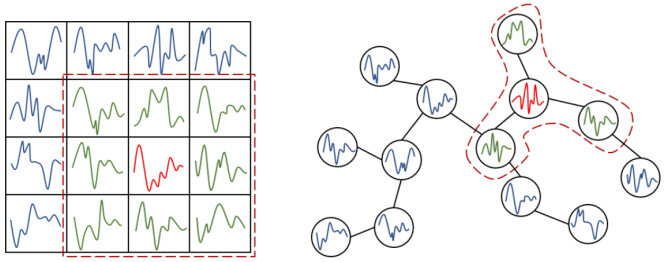


Figure 2.1: Grid-Based Traffic and Graph-Based Traffic (extracted from (R. Jiang et al., 2021))

Categorizing the various dimensions of the Traffic Prediction problem helps to organize this broad task into comprehensible subtasks that will be explored in this dissertation, which will focus on graph-based traffic flow and speed prediction in a freeway context. A diagram that summarizes the dimensions of Traffic Prediction can be seen in Fig. 2.2

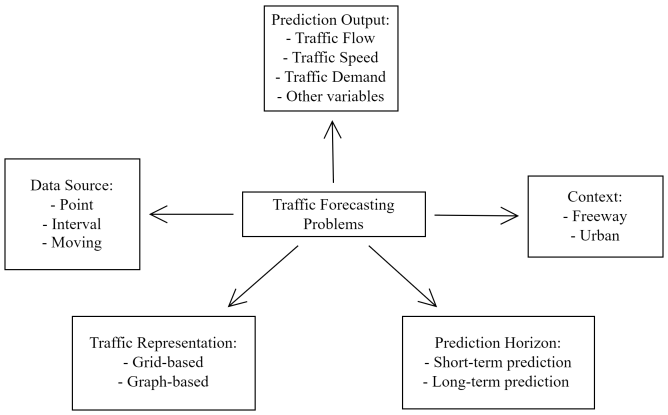
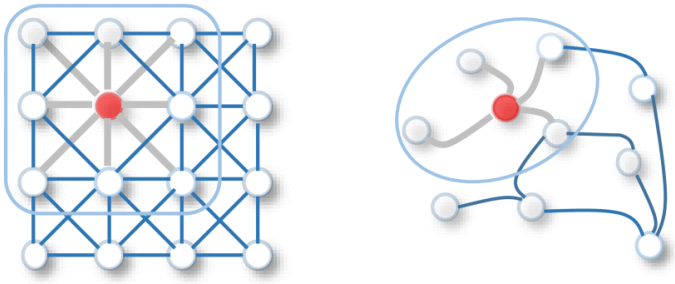


Figure 2.2: Diagram that illustrates the dimension in which traffic prediction problems can be categorized

2.3 Graph Neural Networks

Deep learning has greatly improved the performance of various machine learning tasks in recent years, especially where data is represented in a Euclidean space. However, there is an increasing number of applications where the data is represented as graphs with complex relationships and interdependencies between objects. This complexity of graph data poses significant challenges for traditional machine learning algorithms (Z. Wu et al., 2021b). As a result, there has been a growing interest in developing deep learning approaches for graph data that have culminated in a new concept within neural networks: Graph Neural Networks (GNNs).

GNNs have become the state-of-the-art approach specially designed to operate on data represented as graphs which are a powerful representation for many types of data, including social networks, molecular structures, and in this case, transportation networks (Z. Wu et al., 2021a).



(a) (b)

Figure 2.3: Comparison between a 2-D Convolution (a) and a Graph Convolution (b) (extracted from (Z. Wu et al., 2021b))

In GNNs, the nodes in the graph are typically associated with some feature or attribute, and the edges represent the relationships between the nodes. GNNs learn representations of the nodes and edges in the graph and make predictions about the graph. They are typically composed of multiple layers, each of which takes as input the representations of the nodes and edges from the previous layer and then updates the representations based on the graph structure. The layers of a GNN use various techniques to learn the graph structure, such as aggregation and pooling, which allow the network to propagate information through the graph.

One of the key features of GNNs is their ability to handle graph-structured data with variable sizes and their ability to propagate information through the graph. This allows GNNs to learn from patterns and relationships in the graph that would be difficult or impossible to learn from using traditional neural networks that don't capture this connection, relying on flat data.

2.3.1 General GNN Arquitecture

To better exemplify the architecture of these networks when applied to traffic forecasting problems, a general model will be presented here. The notation used is explained in table 2.1.

The underlying traffic network can be represented as a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{A})$, which can be weighted or unweighted, directed or undirected, depending on specific tasks. Each node represents a traffic object, which can be a sensor, a road segment or a road intersection. $\mathbf{A}$ is the adjacency matrix that contains the topology information of the network. Each entry $\mathbf{a}_{ij}$ in the matrix represents the proximity between a pair of nodes, which can be binary (only capturing if the nodes are connected or not) or a numeric value representing e.g. the length of the road connecting the two nodes (Ye, Zhao, Ye, & Xu, 2022).

$\mathscr{X}_t$ is the feature matrix of the G at time t. The features are usually traffic indicators, such as traffic flow or speed. Given historical indicators of the whole traffic network over past $\mathbf{P}$ time frames, represent as $\mathscr{X} = [\mathscr{X}_1, \cdots, \mathscr{X}_t, \cdots, \mathscr{X}_{\mathbf{P}}]$ aims to predict the future traffic indicators over the next $\mathbf{Q}$ time slices, denoted as $\mathscr{Y} = [\mathscr{Y}_1, \cdots, \mathscr{Y}_t, \cdots, \mathscr{Y}_{\mathbf{Q}}]$, where $\mathscr{Y}_t$ represents output graph with $\mathbf{F_O}$ features at time t. The problem can be illustrated as in Figure 2.4. This structure can be used to predict one traffic indicator or various at the same time.

Table 2.1: Notation used to describe the GNN architecture (adapted from (Ye, Zhao, et al., 2022))

Graph structure elements	
$\mathbf{G}$	Graph
$\mathbf{E}$	Edges of $\mathbf{G}$
$\mathbf{V}$	Vertices of $\mathbf{G}$
$\mathbf{A} = (\mathbf{a}_{ij})_{\mathbf{N}\times\mathbf{N}} \in \mathbf{R}^{\mathbf{N}\times\mathbf{N}}$	Adjacency matrix of graph $\mathbf{G}$
Network parameters	
$\mathbf{N}$	The number of nodes in graph $\mathbf{G}$
$\mathbf{F_I}$	The number of input features
$\mathbf{F_H}$	The number of hidden features
$\mathbf{F_O}$	The number of output features
$\mathbf{P}$	The number of past time slices
$\mathbf{Q}$	The number of future time slices
Spatio-temporal Features	
$\mathscr{X} \in \mathbf{R}^{\mathbf{P}} \times \mathbf{N} \times \mathbf{F_I}$	A series of input graphs composed of $\mathbf{N}$ nodes with $\mathbf{F_I}$ features over $\mathbf{P}$ time slices
$\mathscr{X}_t \in \mathbf{R}^{\mathbf{N}\times\mathbf{F}}\mathbf{I}$	An input graph at time t
$\mathscr{X}_t^i \in \mathbf{R}^{\mathbf{F_I}}$	node i in an input graph at time t
$\mathscr{X}_{t,j} \in \mathbf{R}^{\mathbf{N}}$	the j^{th} feature of an input graph at time t
$\mathscr{X}_{t,j}^i \in \mathbf{R}$	the jth feature of node i in an input graph at time t
$\mathscr{Y} \in \mathbf{R}^{\mathbf{P}\times\mathbf{N}\times\mathbf{F_O}}$	A series of output graphs composed of $\mathbf{N}$ nodes with $\mathbf{F_O}$ features over $\mathbf{P}$ time slices
$\mathscr{Y}_t \in \mathbf{R}^{\mathbf{N}\times\mathbf{F_O}}$	An output graph at time t
$\mathscr{Y}_t^i \in \mathbf{R}^{\mathbf{F_O}}$	node i in an output graph at time t

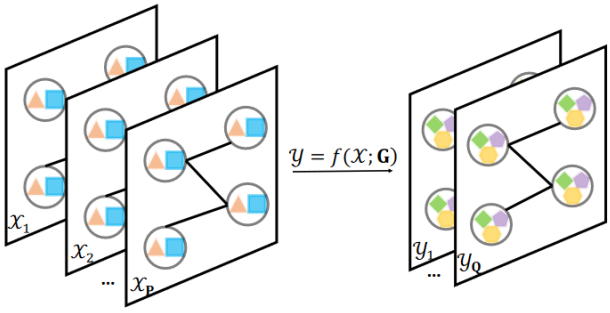


Figure 2.4: Traffic prediction problem formulated as a GNN (extracted from (Ye, Zhao, et al., 2022))

Accordingly to the literature, GNNs can be categorized into four types, namely, recurrent GNNs, convolutional GNNs, graph autoencoders, and spatio-temporal GNNs (Z. Wu et al., 2021a), that will be further explored in Chapter 3.

Because of the spatio-temporal nature of the traffic forecasting problem, the GNNs used in this field can be categorized as spatio-temporal GNNs. However, specific components of the other types of GNNs have also been applied to this problem (W. Jiang & Luo, 2022). Spatio-temporal GNNs are capable of capturing the complex relationships and dependencies between

the different road segments, intersections, and the temporal information, which is essential for accurate traffic forecasting. They can be used to model the temporal dynamics of the traffic on the graph, by using the graph structure to propagate information between nodes and edges, and also the temporal dependencies between the different timestamps. This can be achieved by using temporal convolutions or recurrent layers in the GNN architecture, which allow the network to learn both spatial and temporal representations of the data (Bui, Cho, & Yi, 2022). A general representation of these networks can be seen in Figure 2.5.

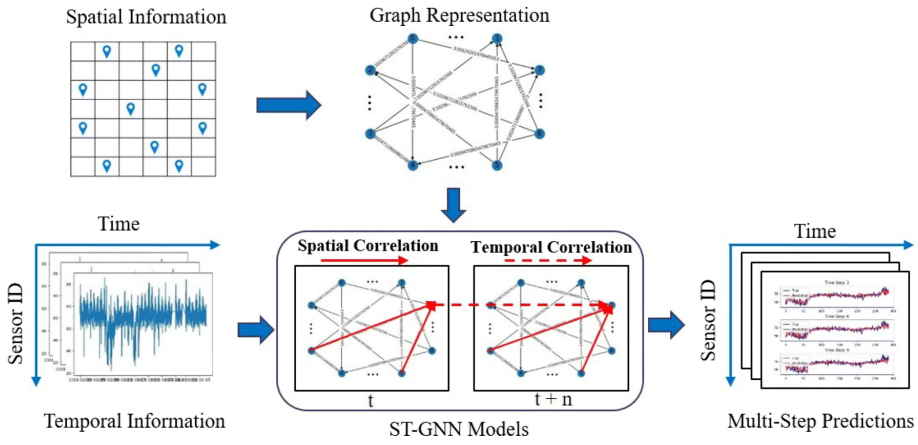


Figure 2.5: Representation of a spatio-temporal GNN (extracted from (Bui et al., 2022))

2.3.1.1 Graph Convolution

One of the most important components of GNNs is the graph convolution. Unlike traditional convolutions, which are used to process grid-structured data, graph convolutions allow for the processing of data that is organized as a graph or a network. In graph convolutional networks (GCNs), each node in the graph is treated as a feature, and the edges of the graph are used to encode relationships between the nodes (B. Yu, Yin, & Zhu, 2018). The goal of graph convolutions is to learn a feature representation of the nodes in a graph.

The basic idea behind graph convolutions is to aggregate information from a node's neighbours in the graph. This is done by defining a function that takes as input the features of a node and its neighbours and outputs a new set of features for the node. This function is applied to all the nodes in the graph in a recurrent manner, allowing for information to flow across the graph and the features of each node to be updated. The parameters of the graph convolution operation are learned during the training process (Sergios Karagiannakos, 2021).

The specific way that graph convolutions are implemented can vary, but typically they involve a combination of matrix multiplication and activation functions. For example, one common approach involves first representing the graph as a weighted adjacency matrix, where the values in the matrix represent the strength of the relationships between nodes. This matrix is then used to

compute a feature representation for each node, which is updated through a series of convolution operations. These operations can also be performed recursively, allowing for a hierarchy of representations to be learned, each representing different levels of abstraction.

2.3.2 Attention Mechanism

The attention mechanism in Graph Neural Networks (GNNs) is a computational technique that allows the model to focus selectively on the most relevant parts of the graph while making predictions (Hamilton, 2020). It allows the model to automatically learn to attend to different parts of the graph, rather than relying on a pre-defined fixed pooling operation that aggregates information from the entire graph. The attention mechanism is implemented as a function that takes as input the node representations and computes a scalar weight for each node. These weights determine the importance of each node in the graph and are used to compute a weighted sum of the node representations, which is used as the graph representation for the next layer (Velickovic et al., 2017). The attention mechanism allows the model to dynamically allocate computational resources to different parts of the graph based on their importance for the prediction task, making it well-suited for complex and dynamic graphs with many inter-dependencies.

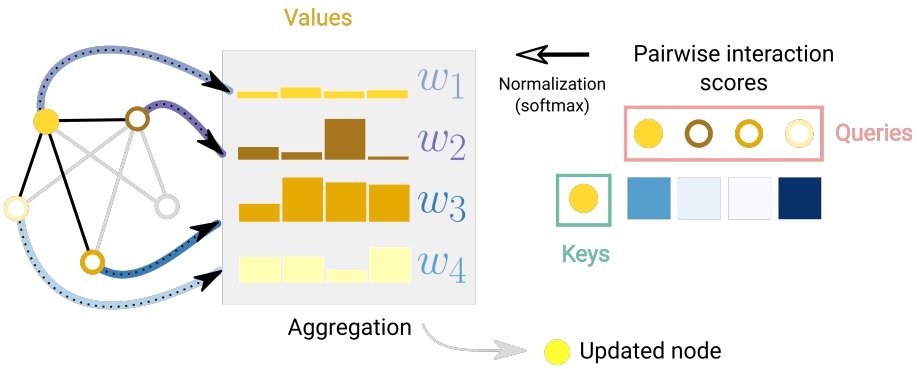


Figure 2.6: Schema of attention over one node with respect to its adjacent nodes. (extracted from (Sanchez-Lengeling et al., 2021))

Note: In the following sections of this dissertation, other specific terms and components related to GNNs and Machine Learning that were not mentioned in this section will be used, so the interested reader is redirected to (Ripley, 1996) for the base knowledge of ML and (Hamilton, 2020)(Z. Liu & Zhou, 2020) for a full overview of GNNs and their structure.

Chapter 3

Related Work

After the context provided by Chapter 2, this section aims to summarize the current state of research in road traffic forecasting, providing a review of the latest technical achievements in this field and highlighting the main challenges that remain for future research in this field and that guide the work developed in this dissertation.

3.1 Traffic Forecasting Techniques

Road traffic forecasting has been a subject of active research for over 40 years (E. I. Vlahogianni, Karlaftis, & Golias, 2014). Initially, most approaches relied on traditional statistical methods for time-series forecasting, such as autoregressive and moving averages models (S. Lee & b. Fambro, 1999) (Smith, Williams, & Keith Oswald, 2002) (Pavlyuk, 2017). However, with the advent of new technologies and platforms for big data processing and the availability of data from multiple sources, the focus has shifted to data-driven procedures (W. Jiang & Luo, 2021). Moreover, with the advent of Machine Learning (ML) and more recently Deep Learning (DL) research has shifted toward these new approaches for tackling traffic forecasting problems (J. Liu, Wu, Qiao, & Li, 2021). Ensemble techniques and hybrid methods have also been applied.

3.1.1 Statistical Methods

Despite being a more traditional approach to this problem, recent approaches based on time series forecasting have still been made. In 2019, Cai et al. developed a noise-immune Kalman filter with maximum correlation entropy for traffic prediction in highways of Amsterdam (Cai et al., 2019). In 2021 Chuvo et al. experimented with traffic prediction with several time-series forecasting techniques (Shuvo, Zubair, Purnota, Hossain, & Hossain, 2021). Despite these recent approaches, the traditional statistical models have certain disadvantages compared to other techniques. They have a very limited ability to handle external factors, spatio-temporal dependencies and missing data (Shi & Du, 2022), and other complexities of real-world traffic data. Still, they are able to make reasonable predictions with small amounts of data (Alsolami, Mehmood, & Albeshri, 2020).

3.2 Regression Algorithms

Regression algorithms have also been applied to this problem. Sun et al. propose a local linear regression model for short-term traffic prediction. The study compares the performance of the proposed model with previous results of nonparametric approaches using 32-day traffic speed data collected on a highway in Houston, Texas. The local linear method is found to be better than the k-NN and kernel methods, as it makes better use of both historical and current data (Sun, Liu, Xiao, He, & Ran, 2003). More recently, (Alam et al., 2019) explored the use of several regression models: Linear Regression, SMO Regression, Multilayer Perceptron, M5P model tree and Random Forest to predict future traffic flow, using real traffic data collected from highways in Porto, Portugal.

3.2.1 Ensemble Learning

Several articles focus on the application of ensemble learning for traffic prediction. In a 2018 study by Chen et al., the authors compared the performance of different ensemble learning methods with traditional time series forecasting methods and found that ensemble methods generally outperformed traditional methods in terms of prediction accuracy (X. Chen, Cai, Liang, & Liu, 2018). Bokaba et al. made a comparative study of the application of several ensemble techniques to predict traffic congestion in a particular case of a South African highway, improving the previously used methods in that case (Bokaba, Doorsamy, & Paul, 2022). Yan et al. proposed an SVR-based ensemble method that was particularly good at dealing with outliers in the dataset and external data, outperforming simpler SVR methods (Yan, Fu, Qi, Yu, & Ye, 2022).

In general, the literature suggests that ensemble learning can be an effective approach for improving the accuracy of traffic flow predictions, especially when data from multiple sources is used, and when it is combined with machine learning models. However, it's worth noting that ensemble learning can also be computationally expensive and require more data and resources to improve performance.

3.2.2 Hybrid Methods

Hybrid methods for traffic forecasting are methods that combine traditional and data-driven approaches to make predictions. These methods typically involve combining the strengths of different models to improve the accuracy of predictions.

(Wang, Li, & Xu, 2017) explored the use of a combination of ARIMA and SVM for traffic flow prediction by conducting a research study, finding that the hybrid methods resulted in improved prediction accuracy. Developing a hybrid model of ARIMA with a Multilayer Perceptron (MLP) and a Recurrent Neural Network (RNN) Rajalakshmi et al. (V & S, 2022) achieved better performance than any of these models achieve on their own. In 2020, Ma et al. (T. Ma, Antoniou, & Toledo, 2020) also developed a model that combined ARIMA with a neural network achieving better results than the models individually.

Overall, hybrid methods have been shown to be effective in improving the accuracy of traffic flow predictions. By combining the strengths of different models, these methods can capture the complexity of traffic patterns and the influence of external factors. It's important to note that while hybrid methods can be effective in improving the accuracy of traffic flow predictions, they may also come with higher computational costs and require more data and resources for optimal performance.

3.2.3 Machine Learning and Deep Learning Techniques

With the increasing amount and diversity of available data and computational power, Machine learning, particularly Deep Neural Networks (DNNs), have emerged as powerful tools for traffic prediction. They can automatically learn complex patterns and relationships in extensive amounts of data and generalise to new, unseen data (Tedjopurnomo, Bao, Zheng, Choudhury, & Qin, 2022).

Among the DNN techniques, Recurrent Neural Networks (RNNs) are the default neural network architecture for handling time-series data. Because of that, they have generally been chosen over Convolutional Neural Networks (CNNs) for traffic prediction tasks (K. Lee, Eo, Jung, Yoon, & Rhee, 2021).

Despite the preference for RNNs, it has been demonstrated that CNNs can also be effective for traffic data because they can better capture its spatial component (K. Lee et al., 2021). This was shown by Ma et al. in 2017 when they were able to get reasonable prediction in traffic speed by modelling the spatio-temporal traffic dynamics as images in a two-dimensional time-space matrix (X. Ma et al., 2017).

As mentioned, research has shown that each of these types of NNs (CNNs and GNNs) are better at capturing one of the two dimensions of the spatio-temporal dynamics of traffic data. CNNs are better at capturing the spatial dimension, while RNNs are better for the temporal dimension. This made the study of hybrid networks that combine a CNN for the spatial part and an RNN for the temporal part a natural choice. A representation of such networks can be seen in Figure 3.1. In 2018, Li et al. constructed a network they called DCRNN (Diffusion Convolutional Recurrent Neural Network) (Li, Yu, Shahabi, & Liu, 2018), capable of capturing spatial and temporal dependencies using diffusion convolution and the sequence-to-sequence learning framework as well as scheduled sampling. In 2019, He et al. also created a model based on this concept which they named STCNN (Spatio-temporal Convolutional Neural Network) (He, Chow, & Zhang, 2019), combining a CNN with Long Short-term (LSTM) Units, achieving state-of-the-art performance at the time of publication. These two approaches are still used as baselines for benchmarking more recent DNN approaches.

Feedforward Neural Networks (FNNs) also play a role in the construction of these hybrid neural networks for traffic prediction, with their main functions being: aggregating the output of subnetworks (Yao et al., 2018) or incorporating external data in the network (Y. Wu & Tan, 2016).

Hybrid DNN methods opened the field for the exploration of other architectures that could capture graph-like structured data, contributing to the emergence of Graph Neural Networks (GNNs).

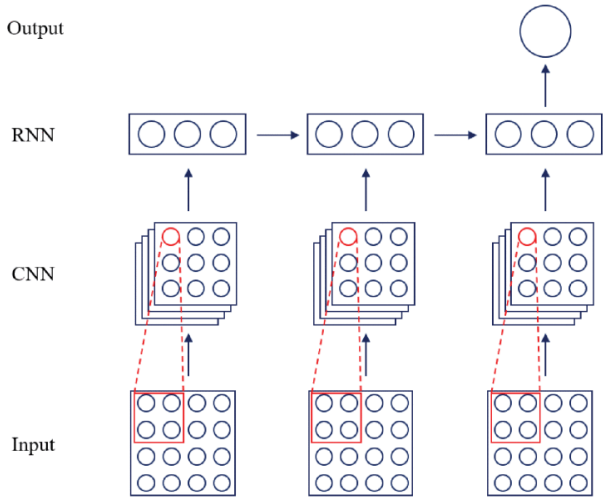


Figure 3.1: Symplifyed representation of a hybrid CNN-RNN model (extracted from (K. Lee et al., 2021))

3.2.4 Graph Neural Networks

As mentioned in Section 2.3, Graph Neural Networks (GNNs) are a recent class of deep learning techniques that are well suited for tasks involving graph data structures, which is the case of traffic prediction. In this problem, it's particularly important to capture not only the spatial graph-like structure of road networks but also the temporal dimension. This notion was first introduced by Yu et al. in 2018, with their Spatio-Temporal Graph Convolutional Neural Network, which integrates graph convolutions and gated temporal convolutions using spatio-temporal convolutional blocks (B. Yu et al., 2018). This model outperformed other state-of-the-art traffic prediction approaches, leading the way for other researchers to explore these networks and serving as a baseline model for comparisons in future works.

In 2019, Chen et al. presented another GNN solution with the adoption of a residual recurrent architecture that can capture graph-based spatial dependencies and temporal dynamics with a special emphasis on capturing periodic temporal correlations (C. Chen et al., 2019). The inclusion of external factors in the models was not considered in these previous approaches, so a new GNN model having these in consideration was proposed by Ge et al.. It uses a graph convolution layer to capture the spatial correlations, followed by multiple layers of casual convolution to learn the temporal dynamics. Then, it considers the influence of social factors and road structure features by using a fusing module (Ge, Li, Liu, & Zhou, 2019). It outperformed approaches such as STGCN and DCRNN while having a shorter training time and fewer parameters. These results were attributed to the choice of a simpler architecture but also to the external data inclusion, hinting towards the fact that these can have a positive impact on predictions. Another approach was proposed by (Guo, Lin, Feng, Song, & Wan, 2019) based on the attention mechanism. This model combines the spatial-temporal attention mechanism and the spatial-temporal convolution, using graph convolutions in the spatial dimension and standard convolutions for the temporal di-

mension, showing improvement from STGCN and other baseline models. The authors pinpointed the inclusion of external data as a relevant future work direction.

A problem that can occur when dealing with road network datasets is that the explicit graph structure does not necessarily reflect the true dependency relation, due to the incomplete connections in the data, so (Z. Wu, Pan, Long, Jiang, & Zhang, 2019) developed Graph WaveNet, a graph-based adaption of WaveNet (van den Oord et al., 2016) that focuses on trying to extract an adaptive adjacency matrix that captures hidden spatial features. Another factor that can influence traffic predictions is the occurrence of incidents, and for that (Xie et al., 2019) developed a model for capturing the impact of traffic incidents on traffic flow and speed, showing how this can impact predictions.

Another proposed architecture is the encoder-decoder, used by (Zheng, Fan, Wang, & Qi, 2019) in GMAN. This architecture features an encoder and a decoder made up of multiple spatio-temporal attention blocks that model the effect of spatio-temporal factors on traffic conditions. The encoder processes the input traffic features while the decoder produces the output sequence. A transform attention layer sits between the encoder and decoder and converts the encoded traffic features into the decoder's input. The transformer attention mechanism models the direct relationships between past and future time steps, helping to mitigate the error propagation issue between prediction time steps. Experimental results demonstrate the superiority of GMAN when compared to other methods, including Graph WaveNet and STGNN, To try to accommodate global network dependencies and changes in the graph structure, (Zhang, Chang, Meng, Xiang, & Pan, 2020) presented an architecture based in Structure Learning Convolution (SLC), which enables extending the traditional CNN to graph domains and learning the underlying graph structure for traffic prediction. The model has two SLC modules to capture the global and local structures respectively. Additionally, Pseudo three Dimensional convolution (P3D) networks are combined with SLC to model the temporal connection. It was tested in 6 different datasets, where it outperformed other state-of-the-art approaches.

Another significant approach was developed by (L. Zhao et al., 2020), combining GCN and GRU networks. The GCN is used to capture the topological structure of the graph to obtain the spatial dependence, and the GRU model is used to capture the dynamic temporal dependence. The experiments made by the authors showed the robustness of the approach when dealing with network perturbations. Another technique that uses GRU was developed by (S. Cao, Wu, Zhang, Li, & Wu, 2022), aiming to address the limitations of conventional graph convolutional networks GCN in mining global spatial correlations. To achieve this, the authors leverage gated recurrent units (GRU) and attention to explore local and global temporal correlations simultaneously.

Trying to tackle the missing data problem in traffic prediction, (Zhong, Suo, Jia, Zhang, & Su, 2021) propose RIHGCN. This method effectively handles missing data through a recurrent imputation process and a heterogeneous graph structure that captures dynamic spatial correlations among nodes in the road network. RIHGCN differs from standard GCN models by creating multiple graphs with different edges, which better capture changing spatial correlations over time. The method integrates data imputation and traffic prediction in a unified framework, optimizing both

objectives simultaneously and avoiding accumulated errors. Experiments show that RIHGCN outperforms existing methods by a significant margin. This work provides an indication that paying special importance to dealing with missing data can bring a positive effect in the predictions.

Addressing limitations of other methods that only model relationships between node pairs and node history information, neglecting node properties, the proposed approach by (Hu, Lin, & Wang, 2022) is a dynamic spatial-temporal graph convolutional network (DSTGCN), which includes a dynamic graph generation module that adaptively fuses geographical proximity and spatial heterogeneity information, and a graph convolution cycle module that captures local temporal dependencies. Experiments on two types of traffic prediction tasks show that DSTGCN outperforms most baseline models, including STGCN, ASTGCN, and GMAN.

In conclusion, Graph Neural Networks have proven to be an effective solution for traffic prediction tasks. Its application has gained significant attention in recent years due to their ability to capture the complex spatial and temporal dependencies of traffic data. The combination of Graph Convolutional Networks (GCNs) and Recurrent Neural Networks (RNNs) has been shown to achieve excellent performance in multi-step traffic flow prediction tasks. Additionally, the introduction of attention mechanisms has further improved prediction results. Furthermore, the inclusion of external factors, such as the time of day, day of the week, and traffic accidents, in the prediction process has been shown to have a significant impact on the accuracy of the prediction results, highlighting the importance of considering the influence of external factors in traffic flow prediction tasks. Table 3.1 contains a list of all the approaches addressed in this section.

3.3 Benchmarking Datasets

As for most prediction tasks, the data used to extract future predictions plays a significant role in the traffic forecasting problem. Common data sources used for traffic prediction include loop detector data, which provides real-time information on traffic volume, speed, and occupancy; GPS data, which provides information on vehicle location, speed, and trajectory; crowdsourced data, which is collected from users through mobile apps or social media platforms; weather data, which provides information on weather conditions; social media data, which provides information on traffic conditions, travel times, incidents and events reported by users; public transportation data, which provides information on the location and schedule of public transportation vehicles; and historical traffic data, which provides information on past traffic conditions, travel times, and incidents. All these sources have their own advantages and limitations, so it is important to evaluate which data sources are most appropriate for a specific traffic prediction problem (W. Jiang & Luo, 2021). The source this dissertation will focus on is inductive loop detectors, sensors embedded in the road surface that detect the presence and passage of vehicles. These detectors provide real-time data on traffic volume, speed, and occupancy, which can be used to estimate traffic conditions, predict travel times, and identify bottlenecks and incidents.

The benefit of utilizing loop detector sensor data for graph-based modelling is that the obtained traffic information can be directly utilized as node attributes, with minimal pre-processing

Table 3.1: Compilation of the relevant existing GNN approaches

Name	**Prediction Horizon**	**Problem**	**Context**	**Tested On**
STGCN: Spatio-Temporal Graph Convolutional Network (B. Yu et al., 2018)	Short-term	Speed	Highway	BJER4, PeMSD7
MRes-RGNN: Gated Residual Recurrent Graph Neural Network (C. Chen et al., 2019)	Short-term, Medium-term	Speed	Highway	PeMS-Bay, METR-LA
GTCN: Temporal Graph Convolutional Network (Ge et al., 2019)	Short-term, Medium-term	Speed	Highway	PeMSD4, PeMSD7
ASTGCN: Attention Based Spatial-Temporal Graph Convolutional Network (Guo et al., 2019)	Short-term, Medium-term	Flow	Highway	PeMsD4, PeMSD8
Graph WaveNet for Deep Spatial-Temporal Graph Modeling (Z. Wu et al., 2019)	Short-term	Speed	Highway	METR-LA, PeMS-Bay
DIGC-Net: Deep Graph Convolutional Network for Incident-driven Traffic Speed Prediction (Xie et al., 2019)	Short-term, Medium-term, Long-term	Speed	Urban	SFO, NYC
GMAN: Graph Multi-Attention Network (Zheng et al., 2019)	Short-term, Medium-term, Long-term	Speed	Urban, Highway	PeMs-Bay, Xiamen
SLCNN: Spatio-Temporal Graph Structure Learning (Zhang et al., 2020)	Short-term	Flow	Highway	METR-LA, PeMS-Bay, PeMs-S
T-GCN: Temporal Graph Convolutional Network (L. Zhao et al., 2020)	Short-term	Speed	Urban, Highway	SZ-Taxi, Los-loop
ST-UNet: Spatio-Temporal U-Network for Graph-structured Time Series Modeling (T. Yu, Yin, & Zhu, 2019)	Short-term, Medium-term	Flow	Highway	METR-LA, PeMS-M, PeMs-L
RIHGCN: Heterogeneous Spatio-Temporal Graph Convolution Network for Traffic Forecasting with Missing Values (Zhong et al., 2021)	Short-term	Speed	Highway	PeMS
GCRAN: Graph Convolutional Recurrent Attention Network (S. Cao et al., 2022)	Short-term, Medium-term	Flow, Speed	Highway	METR-LA, PeMS-Bay
DSTGCN: Dynamic Spatial-Temporal Graph Convolutional Network (Hu et al., 2022)	Short-term	Flow, Speed	Urban, Highway	PEMS-Bay, NE-BJ, PEMSD4, PEMSD8

requirements. However, the sensors are susceptible to hardware failures, resulting in missing or noisy data, which necessitates the use of pre-processing techniques. Additionally, there are some drawbacks to using traffic sensor data for graph-based modelling: these sensors can only be in-

stalled in a limited number of locations due to factors such as installation cost. As a result, only a portion of the road network equipped with traffic sensors can be included in the graph, while the remaining areas would be omitted (W. Jiang & Luo, 2022).

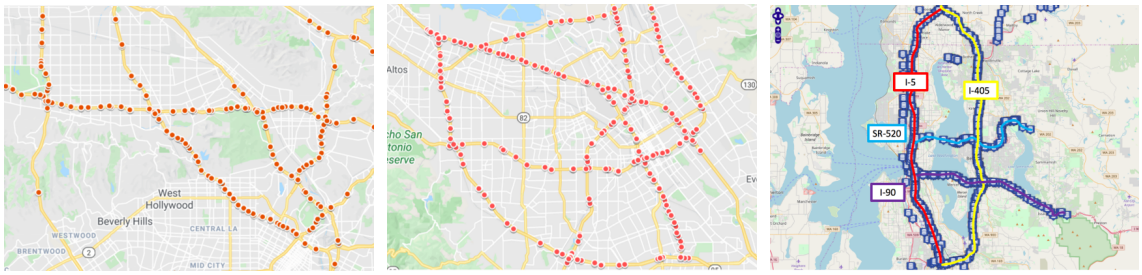
Most GGN-based approaches make their experimental predictions on loop detector datasets, so having open-source datasets that can be used for benchmarking between different approaches is important. Consequently, since the advent of GNNs for traffic prediction, some open-source datasets have become popular and used in different approaches, which are enumerated in Table 3.2 and described as follows.

METR-LA is a dataset that contains traffic speed and volume collected from the highway of the Los Angeles County road network from March 1st to June 30th, 2012, with 207 loop detectors in total, with samples aggregated in 5-minute intervals. It is represented in Fig.3.2a.

Caltrans Performance Measurement System (PeMS) (State of California, 2023) is a large-scale transportation data collection system in California, United States. It is managed by the California Department of Transportation (Caltrans) and provides real-time and historical traffic data from over 18,000 vehicle detector stations on the freeway system. The data is collected using various sensors, including inductive loops, side-fire radar, and magnetometers, and includes information on the total flow, average speed, and direction of travel for each sensor. Several traffic datasets have been extracted from PeMS, namely: PeMS-Bay, which can be seen in Fig.3.2b, contains data from 325 sensors in the Bay Area from January 1st to June 30th, 2017; PeMSD3, which uses 358 sensors in the North Central Area from September 1st to November 30th, 2018; PeMSD4 that contains data from 307 sensors in the San Francisco Bay Area from January 1st to February 28th, 2018; PeMSD7 which uses 883 sensors in the Los Angeles Area from May to June 2012; PeMSD8: that uses 170 sensors in the San Bernardino Area from July to August 2016.

Seattle Loop (Cui, 2023), represented in Fig.3.2c, is a dataset collected on four connected freeways (I-5, I-405, I-90, and SR-520) in the Seattle area, from January 1st to 31st, 2015. It contains the traffic speed data from 323 detectors.

Traffic Speed Guangzhou (X. Chen, Chen, & He, 2018), consists of 214 road segments in Guangzhou, China (mainly urban expressways and arterials) within two months from August 1 to September 30, 2016, in 10-minute intervals.



(a) METR-LA (H. Lu et al., 2020) (b) PeMS-BAY (H. Lu et al., 2020) (c) Seatle Loop (Cui et al., 2018)

Figure 3.2: Image representation of benchmarking datasets

Table 3.2: List of the most popular open-source benchmarking loop-based traffic datasets

Name	**Location**	**Nº of Sensors**	**Aggregation Period**	**Recorded Metrics**	**Time Period**
METR-LA	Los Angeles, CA, USA	207	5 minutes	Speed, Volume	March 1st to June 26th 2012
PeMS-BAY	San Jose, CA, USA	325	5 minutes	Speed, Volume	January 1st to June 30th, 2017
PeMSD3	North Central California, USA	358	5 minutes	Speed, Volume	September 1st to November 30th, 2018
PeMSD4	San Franciso, CA, USA	307	5 minutes	Speed, Volume	January 1st to February 28th, 2018
PeMSD7	Los Angeles, CA, USA	883	5 minutes	Speed, Volume	May to June, 2012
PeMSD8	San Bernardino, CA, USA	170	5 minutes	Speed, Volume	July to August, 2016
Seattle Loop	Seattle, CA, USA	323	5 minutes	Speed, Volume	January 1st to 31st, 2015
Traffic Speed Guangzhou	Guangzhou, China	214	10 minutes	Speed	August 1 to September 30, 2016

3.4 Gap Analysis

In recent years, there has been a growing interest in using graph neural networks (GNNs) for traffic forecasting. Despite the advances made in this area, there are several research to be done and improvement point in current research that need to be addressed in order to further improve the accuracy and applicability of traffic forecasting using GNNs.

One such gap in current research is the handling of missing data, which is a common challenge in many real-world traffic forecasting applications. This can result from missing observations due to equipment failures or data errors or from the unavailability of data for some parts of the network. Some RNN-based models for traffic prediction have explored missing data imputation (Che, Purushotham, Cho, Sontag, & Liu, 2016) (W. Cao et al., 2018). However, despite the prevalence of missing data in traffic forecasting, few GNN-based methods have been proposed to address this issue.

Another gap in the current research is the consideration of external factors (Yin et al., 2022), such as weather conditions or special events, that can have a significant impact on traffic flow (Irawan et al., 2020). These external factors can be difficult to incorporate into GNN models, given the complexity of their interactions with the traffic network. While some efforts have been made to incorporate external factors into GNN models (Ge et al., 2019) (Ye, Xue, & Jiang, 2022), more research is needed to fully understand the impact of these factors on traffic flow and to develop effective methods for incorporating them into traffic prediction models.

Interpretability and explainability are also important areas of concern in the use of GNNs for traffic forecasting (Yuan & Li, 2021) (W. Jiang & Luo, 2022). These models can be complex and difficult to interpret, making it challenging to understand the reasoning behind their predictions (Baldassarre & Azizpour, 2019). This lack of interpretability can limit the trust that stakeholders place in the models and make it difficult to identify areas for improvement. This indicates that there is a need for more research to develop interpretable and explainable GNN-based models for traffic forecasting.

Furthermore, as seen in Table 3.1, most GNN approaches focus on short-term traffic predictions, with intervals from 15 to 60 minutes into the future. Long-term prediction has more complex spatio-temporal dependencies and uncertainty factors (Yin et al., 2022). It is a relevant research direction to make models that are able to deal with longer predictions.

In addition to these issues, there is also a need for further research on the extraction of additional information from the output of GNN-based traffic prediction models. For example, (Xie et al., 2019) have explored the use of GNNs for accident prediction, which is an important application of traffic forecasting. However, there is a need for further research to explore other types of additional information that can be extracted from the output of these models.

Another area that shows a need for further improvement is its general applicability. The different approaches analysed in this literature review were tested in a small number of datasets, often one or two, with little to no focus on exploring how well the models can handle different traffic networks.

In conclusion, while GNNs have shown promising results in the area of traffic forecasting, several gaps in current research need to be addressed in order to further improve their accuracy and applicability. These include the handling of missing data, the consideration of external factors, the development of interpretable and explainable models, and the extraction of additional information from the output of these models. By addressing these gaps, it will be possible to further advance the state of the art in traffic forecasting using GNNs.

Chapter 4

Methodological Approach

This chapter aims to outline the research approach, containing the problem statement and formal definition, a pipeline of the research process, the validation methods and ends with an analysis of the risks that can affect the work's implementation.

4.1 Problem Statement

The main aim of this dissertation is to contribute to the field of traffic forecasting by developing a more accurate and reliable approach to traffic forecasting using GNNs, that can be incorporated into Intelligent Transportation Systems.

To achieve this, the dissertation will explore the following research questions:

- What is the suitable design of a Graph Neural Network for traffic prediction?
- How are the models affected by different missing data handling techniques?
- Can external factors, such as weather and nearby social events, be efficiently incorporated into the GNN models and how do they influence the model's performance?
- Can the model adapt to different real use cases?

4.2 Problem Formalization

The traffic prediction problem can be viewed as a spatio-temporal problem that involves anticipating the future traffic state of the entire road network by using historical data of traffic observations from N sensor stations.

Let the sensor network be defined as a weighted directed graph $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{A})$, following the notation expressed in Table 2.1, where $\mathbf{V}$ is the set of N sensor stations, E is a set of roads connecting the roads where the sensors are present and $\mathbf{A} \in \mathbb{R}^{\mathbf{N}} \times \mathbf{N}$ is the adjacency matrix that contains the distance between sensor stations. Using $\mathbf{x}_t^i \in \mathbb{R}^F$ to denote all features (including traffic speed and the added external data) of node i at time t, then $\mathbf{X}_t = \left(\mathbf{x}_t^1, \mathbf{x}_t^2, \ldots, \mathbf{x}_t^N\right)^T \in \mathbb{R}^{N \times F}$ denotes the

values of all the features of all nodes at time t. $\mathscr{X} = (\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_\tau)^T \in \mathbb{R}^{N \times F \times \tau}$ denotes the value of all the features of all the nodes over τ time slices. In addition, we set $y_t^i = x_t^{f,i} \in \mathbb{R}$ to represent the traffic speed of node i at time t in the future.

So, the problem to be solved becomes that given $\mathscr{X}$, that represents all historical measurements of all the sensors on the network over past τ time slices, being able to predict future traffic states $\mathbf{Y} = \left(\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^N\right)^T \in \mathbb{R}^{N \times T_p}$ of all the nodes on the whole traffic network over the next T_p time slices, where $\mathbf{y}^i = \left(y_{\tau+1}^i, y_{\tau+2}^i, \ldots, y_{\tau+T_p}^i\right) \in \mathbb{R}^{T_p}$ denotes the future traffic state of node i from $\tau + 1$.

4.3 Implementation Pipeline

Figure 4.1 illustrates the methodology followed by this work and summarizes the steps taken to build the GNN traffic prediction model.

The first task consists in retrieving the necessary data, including traffic data, weather and events. This is obtained from various sources and formats. So, the next task is to preprocess the data to make it suitable for the model. This includes cleaning, transforming, and normalizing the data.

Once the data is preprocessed, the model can be trained so it learns to predict the traffic speed based on the input features. The model is then tested using the test data to evaluate its performance using various metrics. Several combinations of parameters will be experimented on, including, for example, adjusting the model's learning rate and optimizer.

Based on the testing results of the different parameters, adjustments are made to the training parameters and new training and consecutively testing takes place.

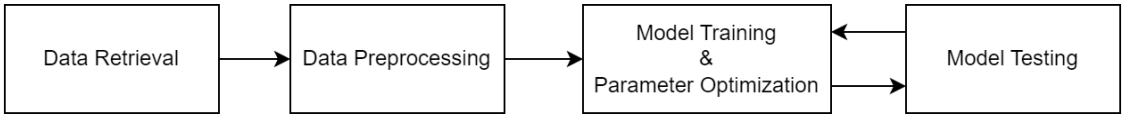


Figure 4.1: Diagram of the proposed implementation pipeline

4.4 Validation

Having a proper validation stage is an important part of the development since it evaluates the performance and assesses how well the model is able to generalize to new data and identify any potential issues.

In this work, the validation is done on three fronts:

- Evaluate the performance on different datasets to evaluate how well the models generalize and adapt to different circumstances;

- Compare the performance of the different iterations (base, with missing data handling and with external factors) of the proposed models to see the impact of these factors in the predictions.
- Compare the performance with baseline approaches to assess how powerful the proposed models are against already existing techniques;

4.4.1 Metrics

The metrics that are used in this validation process are Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). They are mathematically defined as:

$$MAE = \frac{1}{N \times S} \sum_{i=1}^{S} \sum_{j=1}^{N} \left| x_{i,j} - \hat{x}_{i,j} \right| \tag{4.1}$$

$$MAPE = \frac{1}{N \times S} \sum_{i=1}^{S} \sum_{j=1}^{N} \frac{\left| x_{i,j} - \hat{x}_{i,j} \right|}{x_{i,j}} \tag{4.2}$$

$$RMSE = \sqrt{\frac{1}{N \times S} \sum_{i=1}^{S} \sum_{j=1}^{N} \left(x_{i,j} - \hat{x}_{i,j} \right)^2} \tag{4.3}$$

Where $\hat{x}_{i,j}$ is the predicted traffic metric and $x_{i,j}$ is the observed traffic metric, with N being the number of samples in the test set and S the number of sensors. RMSE measures the deviation of the predictions from the actual values, taking into account the magnitude of the error. A lower RMSE value indicates that the model's predictions are closer to the actual values and thus perform better. MAE measures the average magnitude of the errors in a set of predictions, and its units are the same as the predicted target, making it easier to interpret the results. However, as it does not amplify the effect of large errors, it is less sensitive to outliers(Karunasingha, 2022). MAPE provides a measure of the accuracy of the model's predictions as a percentage of the actual values, making it a useful metric for comparing the performance of the model across different datasets. These metrics were chosen based on their systematic presence in the recent literature as the metrics for model performance evaluation (Barros, Araujo, & Rossetti, 2015).

Using these metrics, it is possible to quantify the accuracy of the GNN model's predictions and compare it to the performance of other methods. The validation process results will provide insights into the strengths and limitations of the explored models, allowing for further improvement and refinement. The ultimate goal of the validation process is to demonstrate the effectiveness and feasibility of the explored approaches.

It is important to note that the datasets used contain missing values, and the predictions made for those cases should not be accounted for in the metrics calculation since they would be altering the perceived power of prediction on real values. To avoid that, other works used a technique called masked metrics, in which the missing values are covered by a mask and not used in the metrics computations. This work also uses this technique.

In addition to these metrics, the execution times of the models will also be measured. One model can produce better results than others, but if it takes a much longer time to train, the difference in results might not be relevant enough to justify the added computational cost, so it's important to also extract some conclusions from this.

4.4.2 Baselines

To make a global comparison between the work carried out and research done in this field, several baselines will be used. The choice was based on the baselines chosen by other works, with a focus on time-series methods and the pioneer GNN approaches to the traffic prediction problem. So, the chosen baselines are the following:

- **Historic Average:** This approach models the traffic flow as a seasonal process and uses the weighted average of previous seasons as the prediction (Li et al., 2018). The period used is one week, and the prediction is based on aggregated data from the previous four weeks. For example, the prediction for a given Thursday is the average traffic speeds from the last four Thursdays.
- **VAR - Vector Auto-regressive Model:** It is a multivariate time series model that captures the interactions and dependencies between multiple variables. It represents each variable as a linear combination of its own past values and the past values of other variables. The VAR model generates predictions by estimating coefficient matrices and a constant term. The approach was implemented using the *statsmodel* python package.
- **ARIMA:** Stands for Autoregressive Integrated Moving Average and is a widely used time series model. It is a statistical method that combines autoregressive (AR), differencing (I), and moving average (MA) components to capture the patterns and trends in time series data. The orders used were (3, 0, 1), and the model was implemented using the *statsmodel* python package.
- **DCRNN:** Is a deep learning model created by (Li et al., 2018) specifically designed for traffic prediction tasks. It combines concepts from GCNs and RNNs to capture both spatial and temporal dynamics in traffic data.
- **STGCN:** Is a deep learning model designed by (B. Yu et al., 2018) to capture both spatial and temporal dependencies in spatiotemporal data. It combines GCNs with convolutional and recurrent layers to model spatial and temporal relationships within the data.

4.5 Risk Analysis

Risk analysis is an essential component of any research project as it helps identify potential problems and assess the impact of these problems on the project's outcome. A well-conducted risk

analysis can help minimize potential risks and ensure the research project is completed successfully.

One risk that can take place in this area of research is data availability and quality. Traffic data is crucial for the development and evaluation of the GNN models, but collecting it can be a challenge, especially if the data is proprietary or if there are privacy concerns (Laña et al., 2018). Inaccurate or incomplete data can also negatively impact the performance of the GNN models. To mitigate these risks, this work will utilize benchmarking datasets that have already been used in many other well-documented approaches, which helps ensure the quality and relevance of said datasets.

Another risk is the computational resources required for GNN models. GNNs can be computationally intensive and may require significant computational power and time to train (Yin et al., 2022). This can be a barrier to the practical implementation of GNN models. To address this risk, it is necessary to have access to high-performance computing resources and accommodate the appropriate time for training in the work plan. Since one of the goals of this dissertation is to explore the incorporation of external data in the prediction models, it's necessary to make correct use of those so as not to increase the time and computation power needed to train the models in an unmanageable way.

In addition, there is a risk that the GNN models may not generalize well to new, unseen data due to overfitting the training data. This can lead to poor performance in real-world applications. To mitigate this risk, it is necessary to test the performance of the model in different datasets and adapt it, so it generalizes better.

In conclusion, by considering these potential risks while developing this work, it is possible to minimize their impact and increase the likelihood of achieving high-quality results.

Chapter 5

Implementation

Following the methodology proposition, this chapter aims to explain the implementation details, stage by stage. Starting with the choice of frameworks and technologies for the development, then explaining the data retrieval and preprocessing, followed by the generation of the graph dataset structure. After this data-focused part, the base GNN architecture generation is explained, followed by the additional approaches: handling missing data and adding external data.

5.1 Technologies Used

This work uses Python[1] as the primary programming language, and this choice was motivated by the language's wide popularity and extensive libraries for data analysis and machine learning. Python is a widely-used language for machine learning research due to its large and active community of developers and users, as well as its ease of use and simple syntax. It has a vast ecosystem of libraries and tools designed for machine learning, which provide access to a wide range of algorithms, models, and data processing tools. Python is also a flexible and scalable language that can handle large datasets and complex models, and additionally has interoperability with other languages and tools, allowing to combine their strengths (Raschka, Patterson, & Nolet, 2020). Overall, Python's ease of use, its rich ecosystem of libraries, and its flexibility made it a secure choice for this work.

As for the machine learning framework, PyTorch[2] was chosen due to its popularity, flexibility, and user-friendliness. PyTorch is an open-source machine learning library developed by Meta's AI research team, and has quickly gained popularity in the research community due to its dynamic computational graph and automatic differentiation capabilities. This allows for greater flexibility in model design and a more efficient training process (O'Connor, 2021).

To help with the creation of the graph-structured machine learning methods, PyTorch Geometric[3] was used. It is a library built on top of PyTorch that provides tools for dealing with learning on irregularly structured data, such as graphs. It provides a wide range of graph convolutional

[1] https://www.python.org/
[2] https://pytorch.org/docs/stable/index.html
[3] https://pytorch-geometric.readthedocs.io/en/latest/

layers, graph pooling layers, and graph attention mechanisms that are specifically designed for graph data.

In traffic forecasting, the temporal dynamics of traffic data are critical, so the use of another library, PyTorch Geometric Temporal, was very helpful. PyTorch Geometric Temporal (Rozemberczki, 2023) extends PyTorch Geometric to support temporal graph data and provides a set of temporal graph convolutional layers and temporal pooling layers that can model the temporal dependencies in traffic data. The library also includes utilities for loading and preprocessing temporal graph datasets, making it easy to work with real-world spatiotemporal traffic data.

Using these libraries allows a powerful and flexible toolkit for developing and training deep learning models for traffic forecasting on graph-structured spatiotemporal data. The libraries provide an efficient, easy-to-use, and well-documented framework for deep learning models, taking advantage of the latest advances in deep learning.

5.2 Data Retrieval and Pre-processing

As seen in Chapter 3, the effectiveness of traffic prediction models can benefit from integrating data from multiple sources. In this work, three types of data will be included to predict traffic flow in major metropolitan areas: loop-sensor-based traffic data, weather data, and events data. Combining these three sources of data, the aim is to improve the accuracy and robustness of the traffic prediction model. This section provides a detailed description of each data source, including its collection methodology, processing, and cleaning.

5.2.1 Traffic Data

A pivotal part of the traffic prediction task is the traffic data used for training and testing the models. As seen on Section 3.3, the most used data source for this specific problem comes from loop-detector sensors installed on several points of the road network that measure one or more metrics for a certain time period.

This work aims at using four different datasets, coming from 3 different sources. METR-LA and PeMS-BAY are the two most referenced benchmarking datasets used for GNN-based traffic prediction in the literature, so they will be used to make comparisons easier between the developed approach and the existing ones. Since PeMS-BAY comes from Caltrans Performance Measurement System, it was possible to extract a more recent version of the dataset, to test the use of event data (the availability of events information was only possible for dates starting from 2021). Besides the benchmarking datasets, it was deemed relevant to test the models on another dataset to further evaluate real-world applicability. So, a dataset with similar characteristics that is part of DynamiCITY's research was used. This dataset contains traffic information from "Via de Cintura Interna" (VCI), the ring road of Porto, and was provided by Infraestruturas de Portugal. This dataset is not open-source, but since the work is already going to be tested in two other open-source benchmarking datasets this was not seen as a problem.

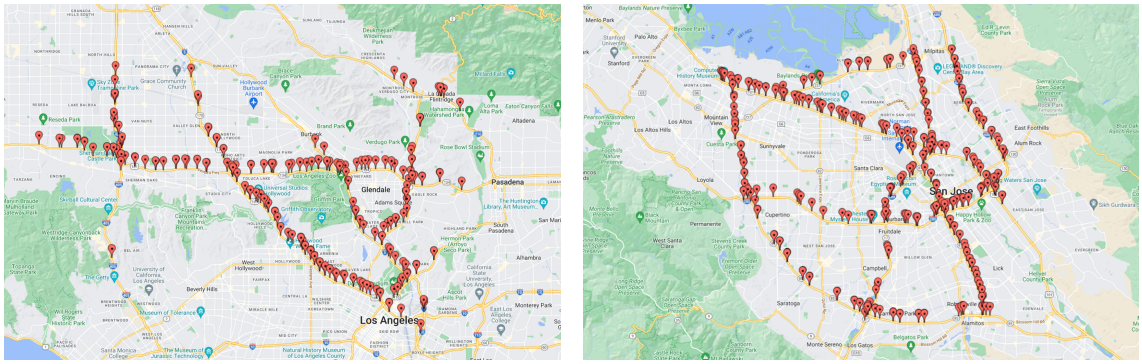
On top of the raw sensor traffic data, in order to be able to generate the graph representation of the traffic, each of the datasets is accompanied by an auxiliary file with the geographic locations of each sensor.

Table 5.1 lists the main characteristics of the used datasets. The particularities and individual pre-processing techniques applied to each of the 3 data sources will be explained in the following subsections.

Table 5.1: Description of each of the different traffic datasets to be used

Name	**Location**	**Number of Sensors**	**Recorded Period**	**Period Length**
METR-LA	Los Angeles, CA, USA	207	March 1st to June 26th 2012	4 months
PeMS-BAY	San Jose, CA, USA	325	January 1st to June 30th, 2017	6 months
PeMS-BAY-2years	San Jose, CA, USA	319	July 1st, 2020 to June 30th, 2022	2 years
VCI	Porto, Portugal	26	January 1st to June 30th, 2015	6 months

5.2.1.1 METR-LA and PeMS-BAY



(a) METR-LA Sensor Location (b) PeMS-Bay Sensor Location

Figure 5.1: Sensor location on the benchmarking datasets

Of the benchmarking datasets found in the literature and described in Section 3.3, PeMS-BAY and METR-LA were the ones chosen to use in this work, because they were the two most referenced. They are both loop-detector-based datasets from highway road networks, with the respective sensor location shown in Fig.5.1. Both datasets also follow the same structure, as illustrated in Fig.5.2: for each time interval of 5 minutes (represented in each row) , there is one value for each sensor (represented by the columns), which is the harmonic average speed of that interval in miles per hour (mph).

AGG_PERIOD_START	400001	400017	400030	400040	400045	400052	400057	400059	400065	...	413877	413878	414284	414694
2017-01-01 00:00:00	71.4	67.8	70.5	67.4	68.8	66.6	66.8	68.0	66.8	...	70.4	68.8	71.1	68.0
2017-01-01 00:05:00	71.6	67.5	70.6	67.5	68.7	66.6	66.8	67.8	66.5	...	70.1	68.4	70.8	67.4
2017-01-01 00:10:00	71.6	67.6	70.2	67.4	68.7	66.1	66.8	67.8	66.2	...	69.8	68.4	70.5	67.9
2017-01-01 00:15:00	71.1	67.5	70.3	68.0	68.5	66.7	66.6	67.7	65.9	...	70.2	68.4	70.8	67.6
2017-01-01 00:20:00	71.7	67.8	70.2	68.1	68.4	66.9	66.1	67.7	66.1	...	70.0	68.4	71.0	67.9

Figure 5.2: METR-LA and PeMS-BAY Dataset Structure, where each column represents a sensor and each row represents a time, with each cell having the harmonic average speed of a sensor in a timeframe

For PeMS-BAY, since the referenced dataset was collected from Caltrans Performance Measurement System (State of California, 2023), which is a large-scale transportation data collection system, it was possible to retrieve a new period of data in addition to the referenced period from January 1st to June 30th, 2017. The new instance of PEMS-BAY data collected dates from the 1st of July 2020 to the 30th of June 2022, with a duration of 2 years in total. There are two reasons for this choice: on one side, it comes from the lack of finding relevant event data for older datasets, so this one can be used to test the incorporation of such factors; on the other side, the bigger temporal window that this dataset has allows for comparing the results when using longer or shorter dataset periods for training the models. This new dataset will henceforth be called PeMS-BAY-2years to distinguish between the two versions. It is relevant to note that the latter version has 319 sensors instead of the original 325 because when retrieving the new data, those sensors' IDs were no longer available. Apart from this, these datasets share the same properties.

5.2.1.2 Via de Cintura Interna - VCI

This traffic dataset was obtained from 26 loop-detector sensors, whose location can be seen in Fig.5.3, installed on Via de Cintura Interna (VCI), a highway in Porto, Portugal. VCI is a ring-like highway that is 21 km long, with two directions separated by a lane separator. The sensors are positioned transversely in the lane, under all pathways in both directions. They collect information about the type and number of vehicles passing through every 5 minutes for the specific segment they are situated in. The historical dataset contains traffic information from 2013, 2014 and 2015, but in order to have a dataset with a similar length to the benchmark ones, only the last six months recorded were used.

Unlike the benchmarking datasets described above, the VCI dataset follows a different structure, with each timestamp having other metrics besides the harmonic average speed value. The meaning of every field in the original VCI dataset can be seen in Table 5.2. Another relevant fact is that this dataset contains missing values, which should be addressed and will pose an additional challenge. For the purpose of this work, it was necessary to change the structure of the dataset to match METR-LA and PeMS-BAY.

To get the original VCI dataset to match the structure of the benchmarking datasets the following procedure is followed:

- Select the needed time period: records between 01/01/2015 to 30/06/2015.

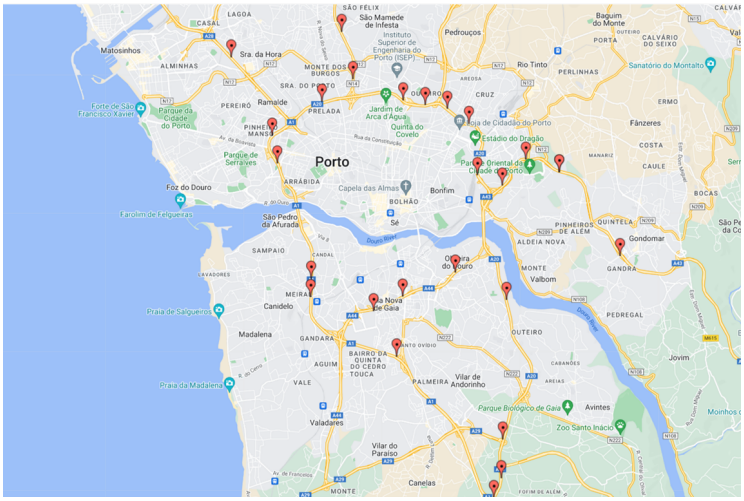


Figure 5.3: VCI Sensor Location

- The original dataset has predictions in the two traffic directions and only one should be studied at a time, so only records with Lane_bunndle_direction equal to "C" were kept.
- Select only the needed fields: Equipment_id, Agg_period_start, Avg_speed_harmonic
- Pivot the dataset table structure to match the structure of PeMS-BAY and METR-LA: Every value of Equipment_id becomes a separate column that for each Agg_period_start holds the speed Avg_speed_harmonic value.
- Avg_speed_harmonic values (originally in km/h) are converted to miles per hour.

5.2.2 Weather Data

5.2.2.1 Description

To have consistent weather data across all traffic datasets, instead of pre-existing datasets, an API was used to retrieve historical weather data for each dataset's locations and time periods. The chosen API was Weatherbit (Weatherbit, 2023a) for its free availability of high-granularity historical data for all locations and time periods needed.

For simplicity, only the closest weather station location to the centre of the road network was picked for each dataset. The smallest frequency of measurements available for historical data is of 15 minutes.

5.2.2.2 Pre-processing

To better fit the needs of this work, the weather data needed to go through some pre-processing steps. First, not all of the dataset fields available[4] through the API were needed for the task at

[4]Full list of available fields: https://www.weatherbit.io/api/historical-weather-subhourly

Table 5.2: Description of each field of the VCI dataset (adapted from (Alam et al., 2017))

Field	**Data Type**	**Description**
Agg_by_lane_bundle_id	Integer	Unique ID of each record
Agg_id	Integer	Street ID
Equipment_id	Integer	Sensor unique ID
Agg_period_start	Datetime	Beginning time of the interval of measurement
Agg_period_len_mins	Integer	Corresponds to the time between measurements made by the sensors. It is invariably of 5 minutes.
Nr_lanes	Integer	Number of lanes in the carriageway segment.
Lane_bundle_direction	String	The direction of the lane where the data was measured. Can be "D" (decreasing) or "C" (increasing)
Total_vol	Integer	Total number of vehicles that went over the sensor
Avg_speed_arithmetic	Float	Arithmetic average speed of the vehicles that went over the sensor
Avg_speed_harmonic	Float	Harmonic average speed of the vehicles that went over the sensor
Avg_length	Float	Average length of the vehicles that went over the sensor
Avg_spacing	Float	Average space between vehicles that went over the sensor
Occupancy	Float	Fraction of the 5-minute interval the sensor was occupied
Light_vehicle_rate	Float	Ratio of light vehicles that went over the sensor
Vol_class_A	Integer	Number of vehicles of class A that went over the sensor
Vol_class_B	Integer	Number of vehicles of class B that went over the sensor
Vol_class_C	Integer	Number of vehicles of class C that went over the sensor
Vol_class_D	Integer	Number of vehicles of class D that went over the sensor
Vol_class_0	Integer	Number of unidentified vehicles that went over the sensor
Axle_class_vol	String	Number of vehicles per vehicle axis number.

hand, so they were not included, such fields were: UV index, solar radiance, atmospheric and sea level pressure, humidity and solar angles. Also, despite being relevant for traffic prediction, the snow-rate field was also removed, given that for the cities concerned in the traffic datasets, its value was always zero. The 'pod' field was converted from string to int, 'n' became 0 and 'd' became '1'.

The final number of fields became 6 out of the original 24 made available by the API. The fields that were kept can be found in Table 5.3.

Table 5.3: Description of the fields from the weather API that were used (adapted from (Weatherbit, 2023b))

Field	**Data Type**	**Units**	**Description**
temp	Float	Celsius	Temperature
wind_spd	Integer	m/s	Wind speed
clouds	Integer	Percentage	Cloud Coverage
pod	String	-	Part of the day. Can be "d" (day) or "n" (night)
visibility	Integer	Km	How far it is possible to see
precip_rate	Float	mm/h	Precipitation Rate

As previously mentioned, the smaller granularity available for the historical data was of 15-minute intervals, while the traffic datasets have 5-minute intervals. So there was the need to match these intervals so the data could be used together. Given the bigger importance of the traffic data over the weather data, it did not make sense to reduce the granularity of the traffic data to 15-minute aggregates, because some information would be lost by doing that. Instead, the weather dataset was modified to have 2 more records between each 15-minute interval by doing linear interpolation of the available records. It is known that this introduces some inaccuracy in the data, but its effects were considered less damaging than reducing the traffic dataset granularity.

5.2.2.3 Automation of weather dataset generation

To allow other researchers to follow the same procedure and generate weather datasets to use with loop-sensor data for traffic prediction tasks, a command line tool that can generate weather datasets with the characteristics as described above for any location and time period was created. It retrieves the data from the Weatherbit API, processes it and saves it to a CSV file.

To use the tool, there's the need to input several parameters, such as a valid Weatherbit API key, the latitude and longitude of the location for which to generate the weather data, as well as the start and end dates for the data and finally the name for the output CSV file. The usage is as follows:

```
1 python weather_tool.py <API_KEY> <LATITUDE> <LONGITUDE> <BEGIN_DATE>
2 <END_DATE> <OUTPUT_FILE>
```

Listing 5.1: Weather tool usage

So, running:

```
weather_tool.py abc123 37.7749 -122.4194 2022-01-01 2022-01-31
    my_weather_data.csv
```

would build weather data for San Francisco, CA from January 1, 2022 to January 31, 2022, using the Weatherbit API key ‘abc123‘, and save it to a file named ‘my_weather_data.csv‘.

5.2.3 Event Data

5.2.3.1 Description

The events dataset was more difficult to find than the previous ones since there is less availability of open-source quality data regarding this matter, and there was the need to find events occurring in specific timeframes in specific locations. There was only one relevant data source found that had the needed information, which was the PredictHQ API (PredictHQ, 2023a). The PredictHQ API aggregates and enriches event data from various sources to provide a comprehensive and up-to-date view of events happening around the world. It includes information such as event type, location, time, duration, and attendance. Unfortunately, it is only possible to retrieve the historic events data for free if they are less than a year in the past, counting from the moment of the API call, so it created the need for having a more recent traffic dataset, as previously explained.

The choice of events to retrieve from the API was based on the following criteria:

- The location of the event needs to be within a 20 km radius of the road network;
- The event’s occurrence needs to be between the recorded time period of the traffic dataset;
- The attendance should be of at least 1000 people, this was deemed a reasonable threshold between small gatherings and events that have the potential to significantly affect the way people usually move in the city;
- The event belongs to one of the following categories: conferences, expos, concerts, festivals, performing arts, sports, community, public holidays, school holidays, severe weather and disasters. The excluded categories were non-attendance-based events, such as daylight savings, which were considered to not have an impact on mobility.

5.2.3.2 Pre-processing

In terms of pre-processing, the data provided by the API did not need much work to be ready for use in this work, so the pre-processing procedure was simple. One of the required steps consisted in removing the variables that would not be relevant to this use case. The event records directly retrieved contained 32 different fields (PredictHQ, 2023b), with information such as if the event was private or public, if it had been rescheduled or postponed when it was announced, and many others that don’t need to be included, so they were removed. In addition, the location of the event was originally expressed as a complex datatype with several sub-fields, so it was changed in order

to only have two separate fields of longitude and latitude. Table 5.4 shows the fields and respective descriptions of the processed dataset. After these pre-processing steps, the final events dataset for PeMS-BAY-2years is comprised of 394 events, with the category distribution as shown in Figure 5.4.

Table 5.4: Description of each field of the Events dataset

Field	Data Type	Description
category	String	Type of event (concert, holiday, sports, etc...)
start	Datetime	Timestamp of the start of the event
end	Datetime	Timestamp of the end of the event
lat	Float	Latitude of the event's location
long	Float	Longitude of the event's location
attendance	Integer	Event's attendance

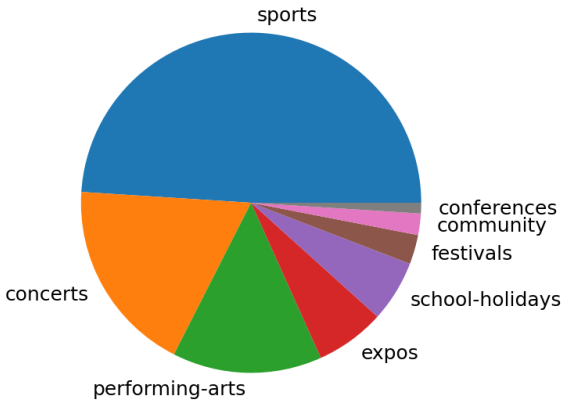


Figure 5.4: Category distribution of the events retrieved for PeMS-BAY-2years

5.2.3.3 Automation of events dataset generation

To allow other researchers to follow the same procedure and generate events datasets to use with loop-sensor data for traffic prediction tasks, a command line tool that can generate events datasets with the characteristics as described above for any location and time period, as long as they are from less than one year in the past from the moment the API call is being made, was created. It retrieves the data from the PredictHQ API, processes it and saves it to a CSV file.

To use the tool, there's the need to input several parameters, such as valid PredictHQ API client ID, API secret and access token, the latitude and longitude of the 'epicentre' for which to generate the data, as well as the start and end dates for the data in and finally the name for the output CSV file. The usage is as follows:

```
1 python event_search.py  <CLIENT_ID> <API_SECRET> <ACCESS_TOKEN> <LATITUDE>
2 <LONGITUDE>  <RADIUS> <BEGIN_DATE>  <END_DATE>   <OUTPUT_FILE>
```

Listing 5.2: Events tool usage

So, for example, running:

```
python events_tool.py 1234 secret token 40.7128 -74.0060 10
 ↪  2023-03-01 2023-03-31 nyc_events.csv
```

would build a dataset with events in New York City within a radius of 10 kilometres between March 1, 2023 and March 31, 2023, and export the results to a file called nyc_events.csv

5.3 Graph Dataset Generation

After the retrieval and pre-processing of data, the next step is constructing the input to feed the GNNs from the original data.

Since this work will be tested on several different datasets, one of the goals was being able to create a uniform way of assembling the GNN input data from any given dataset, both to ease the modular development of this work and to further increase its future usability for other use cases, i.e. other road networks or different recorded time periods. For this, it was important to determine the information needed to be able to generate the input data, which was ultimately narrowed down to this set of files :

- A csv file that represents the speed values per timestamp in every sensor, with the structure shown in Fig.5.2, henceforth called traffic data;
- A csv file with information about the coordinates of the sensors, with the structure as shown in Figure 5.5.

	sensor_id	latitude	longitude
0	121725	41.077408	-8.577612
1	121726	41.082057	-8.575278
2	121727	41.091385	-8.574568
3	121729	41.124692	-8.573643
4	121730	41.154599	-8.582563
5	121737	41.166756	-8.585464
6	121738	41.170352	-8.592347

Figure 5.5: Structure of the file with the sensors' location

Given these files, it is possible to generate the input data for the GNN models for any given road network and time period by following the procedures shown in Fig.5.6, which will be further described in the next subsections.

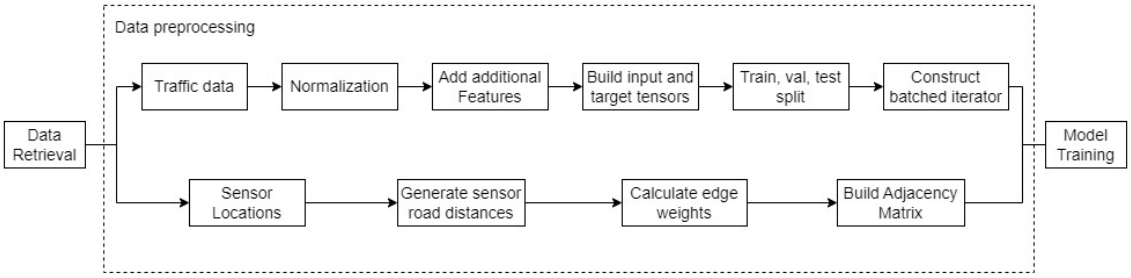


Figure 5.6: Description of the graph dataset generation

It is important to note that this data generation phase is adapted to add the additional features of handling missing data and external data incorporation, which will be described in Section 5.5 and Section 5.6, respectively.

5.3.1 Adjacency Matrix Generation

5.3.1.1 Calculating Road Distance Between Sensors

Having the information from the sensors' location is not enough to extract all the spatial information of the road network since the road structure and direction of traffic flow also play a part. Having an adjacency matrix that represents the distances between the sensors is needed. For PeMS-BAY and METR-LA datasets, the distance between sensors was made available by the developers of the DCRNN approach in their GitHub repository (Li, Mensi, & Yu, 2023). However, for the VCI dataset, it was not available, so there was a need to create it. It may seem trivial to obtain these distances by calculating them from the distance between the coordinates, but this would be a very rough estimation of the true adjacency of the road distance (i.e. the distance that a car would need to travel from an origin to a destination) between the sensors. So, an alternative was found by using Openrouteservice[5], which is a free-of-charge and open-source API, which provides road distances between points. Like this, it was possible to create a way to build adjacency matrices for this particular VCI dataset and, in general, for other datasets that might be used in the future.

5.3.1.2 Calculating Edge Weights

Having the road distance between sensors, it is now possible to calculate the weight of the graph's edges. Instead of using the raw distance values between sensors to fill the adjacency matrix, this work follows the technique used in DCRNN(Li et al., 2018) and STGCN(B. Yu et al., 2018), in which the weighted adjacency matrix W is filled using the Thresholded Gaussian Kernel function (Shuman, Narang, Frossard, Ortega, & Vandergheynst, 2013):

[5] https://openrouteservice.org/dev/#/api-docs/v2/matrix

$$W_{v_i,v_j} = \begin{cases} \exp\left(-\frac{[\text{dist}(v_i,v_j)]^2}{2\sigma^2}\right) & \text{if } \text{dist}(v_i,v_j) \leq \kappa \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

where W_{v_i,v_j} is the edge weight between sensor v_i and sensor v_j, $\text{dist}(v_i,v_j)$ represents the road network distance from v_i to v_j, σ is the standard deviation of distances and κ is a threshold for sparsity, meaning that sensors with a weight lower than the threshold are set to 0. This is especially helpful for bigger networks where sensors very far from each other have almost no interaction, and therefore their connection can be ignored by the network model to speed up calculations. The threshold k was set to 0.1 as it was done by the approaches mentioned above.

5.3.2 Traffic Speed Normalization

Normalizing data for neural networks is an important step to take since it helps optimization algorithms by balancing the scales of input features, preventing imbalanced gradients. It accelerates the convergence process by enabling quicker identification of the optimal solution and enhances generalization by reducing sensitivity to variations in absolute magnitude and focusing on underlying patterns within the data (Bhanja & Das, 2019).

The normalization method chosen was Z-score normalization since it is more robust with outliers than min-max normalization and doesn't restrict the data to a given range.

5.3.3 Additional Features

On top of the speed sensor readings, adding additional temporal features that could help capture the temporal dependencies between the data was deemed important. So, considering the base traffic dataset as an array with shape $[T,S]$, where T is the number of timestamps and S the number of sensors, in this step, another dimension is added so for each sensor and timestamp instead of only having the traffic speed information, there can be other features, making the shape $[T,S,F]$, where F is the number of features (traffic speed included).

The explored features were:

- Time in day: every five-minute interval of the day gets a different value between 0 and 1 sequentially, so the same time on different days shares this feature to try to capture daily patterns.
- Day in week: all records that are from the same day of the week get the same value to try to capture weekly patterns.
- Hour in day: all records that are from the same hour of the day get the same value.
- Is weekend: all records that belong to weekends get 1 and others get 0

The goal here is to train the models with different combinations of these features to see if they help the results. Here only temporal features extractable from the traffic dataset itself were explained. Data from other sources are explored in Section 5.6.

5.3.4 Building Inputs and Targets

The models explored throughout this work are sequence-to-sequence models: for a given timestamp, the observed feature values of that timestamp plus an additional number of immediately preceding values are taken as historical features to predict x future speed values. From now on, the number of historical features and the number of future prediction horizons will be called '$num\_historical\_features$' and '$num\_prediction\_horizons$', respectively.

Having this in mind, when creating the input and target tensors for the model, they will have the following shapes $[T, S, F, num\_historical\_features]$ and $[T, S, num\_prediction\_horizons]$.

The input tensor has four dimensions, meaning that the model takes as input for each sensor in each timestamp '$num\_historical\_features$' of historical features preceding the given timestamp. The output tensor has three dimensions meaning that the model for each sensor in each timestamp generates '$num\_prediction\_horizons$' number of predictions into the future, each with 5 minutes between them. The tensor only has the speed information and not the additional features present in the input tensor since these are not being predicted and are only used to generate the speed predictions.

After having the dataset in this final form, it can be split for the train, validation and test.

5.3.5 Train, Validation and Test Splitting

In terms of splitting the dataset, in this case, since Neural Networks are used, and they need to have their hyperparameters tuned during training, on top of a train and test sets, a validation set is also needed.

The splits are done in the time dimension, i.e. each subset contains a portion of the timestamps of the dataset but the entire set of sensors.

In most cases, this split should be done by randomly selecting which samples go into each subset. However, since the data in question has a temporal aspect, this choice needs to be reconsidered. Time-series data has a chronological order and, therefore, some form of temporal dependencies. This is different from many other types of data, where the observations are usually considered to be independent of each other. By randomly splitting time-series data, some "future" data would likely be in the training set and some "past" data in the test set. This violates the temporal order and dependency of the data, and it results in "data leakage", where the model has access to future information it wouldn't have at prediction time (Jensen, 2019). This can lead to overly optimistic performance estimates during training and validation stages, which might not hold when the model is applied to real-world, future data.

When splitting the data chronologically (i.e., the training set consists of all data up to a certain time point, followed by validation and then the test set), the temporal order of the data is maintained, which is a more realistic approach and represents how the model will be used in practice: trained on past data and used to make predictions for future data. Furthermore, chronological splitting allows the model to capture possible trends, seasonality, and other temporal dynamics in the training data, which might be relevant for future predictions.

5.3.6 Dataloader

After having the data split, the last step is to use PyTorch's 'DataLoader' method to divide inputs and targets into batches of a specified size, allowing for more efficient processing. It makes an iterable object from the batches of inputs and targets, allowing iteration over the batches for training and test loops.

The data in this form, together with the adjacency matrix generated, are what is fed the model to obtain future predictions.

5.4 Base Graph Neural Network Architecture

In this study, the capabilities of PyTorch Geometric and PyTorch Geometric Temporal libraries will be used to adapt GNN architectures for spatiotemporal traffic prediction. These libraries provide a comprehensive set of building blocks and tools that facilitate the design and implementation of GNNs. This part of the work aims to discover which architecture achieves superior performance across various datasets.

The process begins with the exploration of candidate architectures with a set of base hyperparameters configuration, on all datasets. This serves as a primary evaluation of the performance of the different models, allowing the identification of the most promising architectures. Once the two top-performing architectures have been identified, the next step is to evaluate their performance with other hyperparameters.

Upon selecting the optimal GNN architecture and the set of optimal hyperparameters, the aim is to try to enhance the performance further by incorporating external data sources and implementing strategies to handle missing data.

5.4.1 Layers

The models explored in this work follow the structure illustrated in Fig.5.7. The data goes through N Spatio-temporal blocks made up of a spatial graph mechanism followed by a temporal mechanism. Several mechanisms were tried, as enumerated in Table 5.5. The number of blocks is a parameter that was explored in the empirical experiments. After the spatio-temporal blocks, the data goes through a fully connected layer, followed by ReLU activation, to result in the final network output.

Table 5.5: Enumeration of the temporal and spatial mechanisms that were explored

Spatial Graph Mechanisms	
GATConv	Graph Attention Convolution
GCNConv	Graph Convolution
ChebConv	Chebyshev Spectral Graph Convolution
Temporal Mechanisms	
GRU	Gated Recurrent Unit
CNN	Convolutional Neural Network
GRU + Attention	GRU + Temporal Attention
CNN + Attention	CNN + Temporal Attention

5.4.1.1 Spatial Graph Mechanism

This part of the network is responsible for spatial dependency modelling, capturing the relationship between neighbouring sensors in the network.

The mechanisms that were explored are the following:

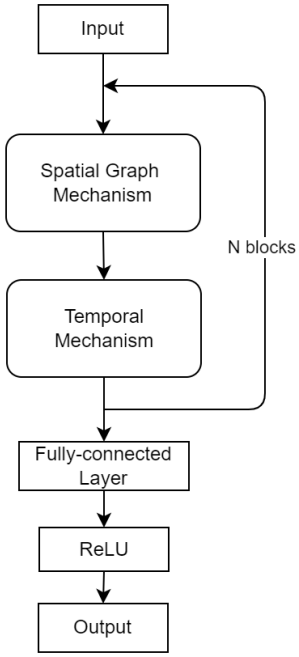


Figure 5.7: Structure of the GNN layers

- **GAT - Graph Attention Network:** GAT leverages attention mechanisms to model the relationships between nodes in a graph. GAT was introduced in (Velickovic et al., 2017) to capture the importance of neighbouring nodes when aggregating information during message-passing iterations. In GAT, each node in the graph is associated with a learnable attention mechanism. Attention allows the model to assign different weights to the neighbouring nodes based on their importance in the context of the target node. This mechanism works with multi-head attention, where k different mechanisms run in parallel, originating k independent outputs that are then concatenated and linearly transformed. This parameter k is set to 3 in this work, as was done in the original paper.

- **GCN - Graph Convolution Network:** The graph convolution operation was introduced in (Kipf & Welling, 2017). It updates a node's features by aggregating its neighbouring nodes' features. This process is repeated for several layers, each time considering a larger neighbourhood, which allows the network to learn more global features of the graph.

- **ChebConv - Chebyshev Spectral Graph Convolution:** This graph operation was introduced in (Defferrard, Bresson, & Vandergheynst, 2016), and it is a variation of GCN designed to approximate spectral graph convolutions by using Chebyshev polynomials. It operates in the graphs' spectral domain, leveraging the Laplacian matrix's eigenvalues and eigenvectors. By truncating the expansion of the Chebyshev polynomials, it efficiently computes convolution on graph signals without the need for costly eigenvalue decomposition. This operator allows for the propagation of information and capturing of spatial dependencies in graph data.

5.4.1.2 Temporal Mechanism

This part of the network is responsible for temporal dependency modelling, capturing the influence of past sensor readings with more recent ones.

The mechanisms that were explored are the following:

- **Gated Recurrent Unit:** The most commonly used neural network model for processing sequence data (i.e. temporal data) are RNNs. However, traditional RNNs have long-term prediction limitations due to gradient disappearance and explosion. To address these problems, variants of RNNs such as the LSTM and GRU have been developed. As LSTM has a more complex structure and longer training time, while the GRU model has a simpler structure and faster training ability, the GRU model is chosen to obtain temporal dependence from traffic data in some works, such as (Li et al., 2018) (L. Zhao et al., 2020). The GRU model uses a hidden state from the previous time step and the current traffic information as inputs to predict the traffic status at the current time step. This allows the model to capture the current traffic information while retaining the changing trends of historical traffic information and capturing temporal dependence.

- **CNN:** Despite RNNs being more commonly used in temporal data, CNNs are also used to model this part of the data in works such as (B. Yu et al., 2018) due to their faster training times, simpler structures, and lack of dependency on previous steps. Inspired by (Gehring, Auli, Grangier, Yarats, & Dauphin, 2017) the authors propose a method that employs entire convolutional structures on the time axis to capture the dynamic behaviours of traffic flows. This design allows for parallel and controllable training procedures. The temporal convolutional layer comprises a one-dimensional causal convolution with a width-Kt kernel, followed by Gated Linear Units (GLU) for non-linearity. This process essentially maps the input (which could be seen as a multi-channel time series) to a single output element via a convolutional kernel.

- **Gated Recurrent Unit + Attention:** This mechanism was used in (Zhu, Song, Zhao, & Li, 2020), has an extension of their previous work (L. Zhao et al., 2020), that after passing through GRU, the hidden states are inputted into an attention model to determine the context vector that covers the global traffic variation information in time.

- **CNN + Attention:** This mechanism was introduced in (Guo et al., 2019), and it consists of combining a CNN with an attention mechanism to capture the temporal dependencies.

5.4.2 Network Hyperparameters

One of the challenges of this work is to identify the optimal combination of parameters that yield the best performance for the specific network architectures that are being explored. By systematically investigating the impact of different parameter settings, the aim is to understand their influence on the overall performance and generalization capabilities of the GNN models.

The importance of selecting appropriate network parameters cannot be overstated, as it directly affects the model's capacity to learn and adapt to the underlying data patterns. A well-tuned GNN model can significantly improve accuracy and robustness while reducing the risk of overfitting and ensuring efficient training.

This section describes the parameters that were explored in this work. It is important to note that the parameter exploration needs to be kept to a reasonably minimal size not to create a combination too large to be experimented on.

- **Learning Rate:** The learning rate is a crucial hyperparameter determining the step size during optimization. A suitable learning rate can significantly improve the model's convergence speed and overall performance. However, selecting an appropriate learning rate can be challenging, as it often depends on the problem domain, data distribution, and model architecture. While a larger learning rate leads to a premature, sub-optimal solution, a smaller one will converge the optimisation process slowly (K. Zhou, Liu, Duan, & Hu, 2022). Already existing approaches used either 0.01 or 0.001 as a base learning rate. In this work, the base learning rate was set to 0.01 on the first tests done, but on top of those, some other experiments with other learning rates, including 0.001, were also carried out.

- **Training Epochs:** The number of training epochs is a key hyperparameter that determines how many times the model will update its parameters based on the training data. Balancing the number of epochs is essential to avoid underfitting, where the model fails to capture the underlying patterns in the data, and overfitting, where the model becomes overly specialized to the training data and performs poorly on unseen instances. Recent GNN studies (Li et al., 2018)(B. Yu et al., 2018)(Guo et al., 2019)(L. Zhao et al., 2020) indicate the use of 100 epochs for training their GNN models, so the same is applied in this work.

  An important detail to note is that the model that gets saved from training to be applied to the test data and evaluated is not the model that results from the last epoch. but rather the model from the epoch that got the lowest loss in the validation data. This means that in the training loop the current validation loss is always checked against the best until that point , and if it is even lower, the model gets saved.

- **Loss Function:** Most GNN works use MSE (Li et al., 2018) (B. Yu et al., 2018) (T. Yu et al., 2019) loss, so this was also applied in this work. It is important to note that since these works don't do any missing data pre-processing when calculating the loss function, they use a variation of the original definitions, where the loss is only calculated based on non-missing values by applying a mask to null values. These are often called masked loss functions.

- **Optimizer:** The optimizer is the algorithm that adjusts the network's weights and biases to minimize the loss function. It does this by computing the gradients of the loss function according to the parameters and updating them iteratively. The optimizer's goal is to find the best parameter values that reduce the loss, improving the network's predictions(Lau, 2017). Previous works with GNN mostly use the Adam optimizer. In addition to Adam,

other optimizers, AdamW and AdaGrad, will be used in the experiments to see if they can produce better results.

- **Train, Validation and Test proportions:** This work used 80% of data for training, 10% for validation and 20% for testing.
- **Number of input features and output targets:** As mentioned in Subsection5.3.4, when building the input data, n historic features are taken and the output contains x predictions. These numbers are configurable and so different combinations of input features and output targets were explored, but the default value is 12 inputs and 12 outputs, meaning one hour's worth of data is taken as input and the model generates 12 predictions, for 5 minutes after the origin timestamp, 10 minutes after, 15 minutes after and so on until one hour.
- **Number of spatio-temporal blocks:** Other techniques that use a similar consecutive spatio-temporal blocks approach show better results with 2 blocks (Guo et al., 2019) (L. Zhao et al., 2020), so this was the number used in the experiments of this work.

5.5 Handling Missing Data

As previously explained, the existing techniques regarding traffic speed prediction with GNNs do not attempt to handle the missing data present in the datasets they use but instead mask them when calculating loss and evaluation metrics. So in this work, it was explored if dealing with the missing data can positively impact the predictions made by the models. But first, it is important to explore the characteristics and quantity of the missing data present in the datasets.

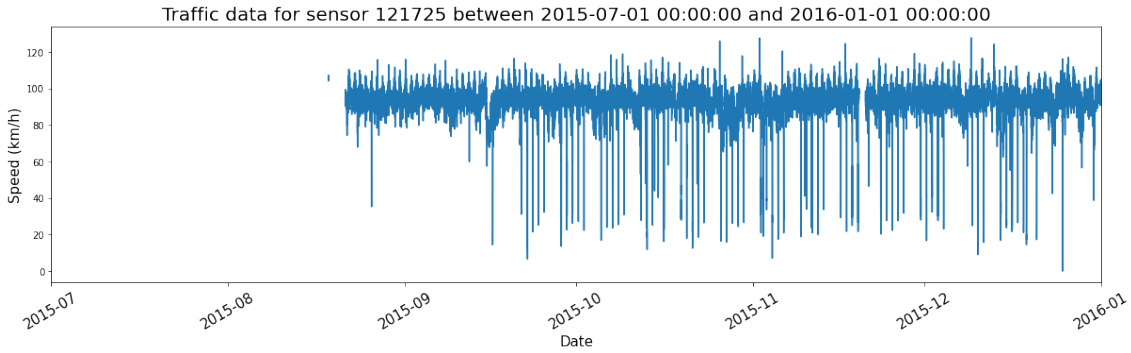
5.5.1 Missing data Occurrence and Characteristics

After analysing the datasets, it was possible to observe different types of occurrences of missing data. In the case of VCI, some sensors showed no readings for long periods of time, sometimes for more than a month, as shown by plotting the entire set of sensor readings in Fig.5.8a, in which the first 1.5 months are not present. This can mean that the sensor in question was broken and it took that amount of time for it to be replaced or fixed.

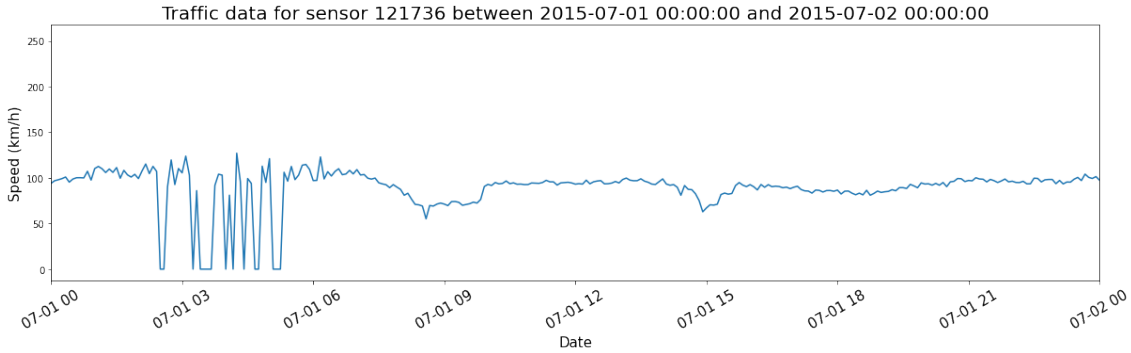
On top of these blocks of missing data, there also happen sparse missing data, when in the middle of valid sensor values are missing values or zeros. These can happen by sensor failure but can have a different reason: the sensors record the average speed of vehicles passing through them in the given time period, so if for a period of 5 minutes, no vehicle passes through, then the sensor will register 0 as the speed value for that period, which is also not a valid reading. An example of this can be seen in Fig.5.8b, plotting only one day of a sensor's readings, and there are several 0 values during the late night hours.

In terms of the number of missing values, the four datasets used in this work have distinct proportions, as shown in Table 5.6. PeMS-BAY and PeMS-BAY-2years are significantly less

affected by missing values than the others. METR-LA, despite not having blocks of null values in its sensors, has a significant amount of sparse zeros, more than in the VCI dataset.



(a) Block of missing data



(b) Sparse missing data in the form of zeros

Figure 5.8: Types of occurring missing data in the sensors (VCI dataset)

Table 5.6: Presence of missing values in the dataset

	Ratio of null values	**Ratio of zeros**	**Total ratio of missing values**
METR-LA	0%	8.11%	8.11%
PeMS-BAY	0%	0.03%	0.03%
VCI	4.09%	0.24%	4.33%
PeMS-BAY-2years	0.01%	0%	0.01%

5.5.2 Missing Data Handling Techniques

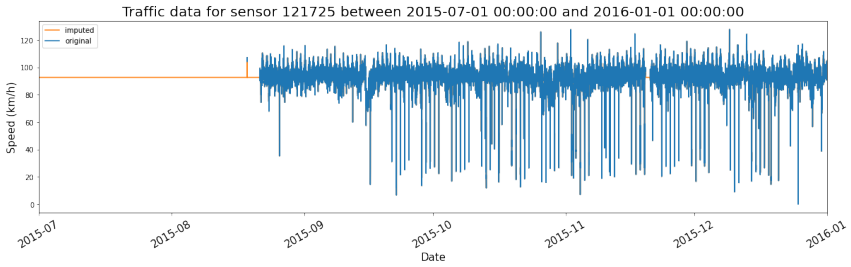
After identifying the missing data, the next step is to find appropriate ways of dealing with them, keeping in mind the special characteristics of the data at hand.

The problem as it is represented in this work, regards the traffic data as a spatio-temporal dataset, with the readings of one sensor having a correlation with other sensors, so removing the missing data is not an option in this case. Instead, data imputation techniques will be explored. For the choice of these techniques, since most of them work sensor by sensor (not considering the

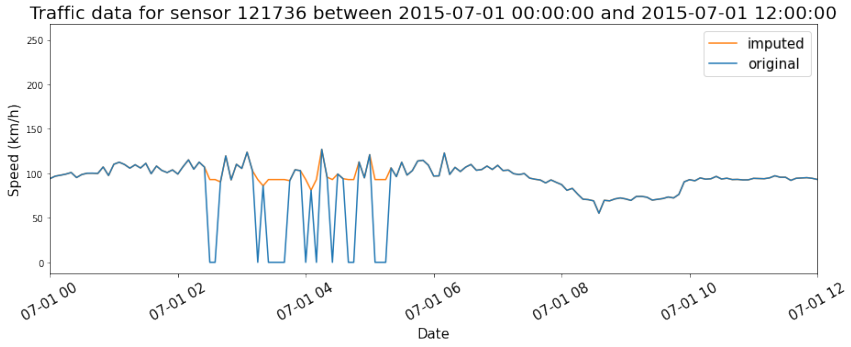
network correlations) and each sensor individually can be considered a time series of traffic speed records, the techniques chosen to apply should be ones used in time series data.

5.5.2.1 Mean

This is a simple and commonly used technique for handling missing data. The missing values are replaced with the mean value of the sensor readings. Used with time series data it has limitations since it does not account for temporal dependencies between sensor readings. The plotted values of the imputation in a sample of data from the VCI dataset can be seen in Fig.5.9a for a block of missing values and Fig.5.9b for sparse missing data.



(a) Block imputation



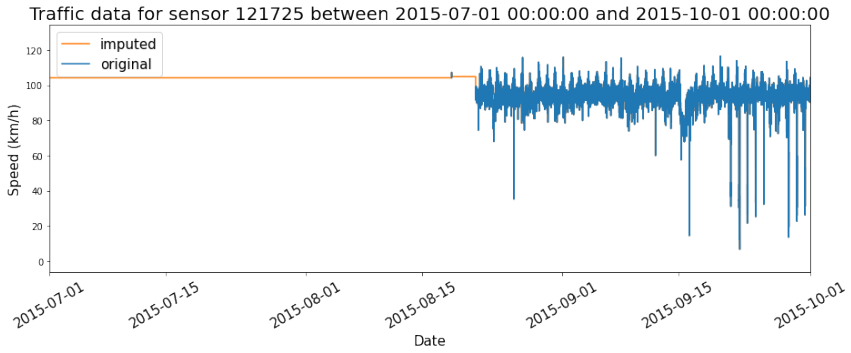
(b) Sparse imputation

Figure 5.9: Comparison between original data and after mean imputation in VCI

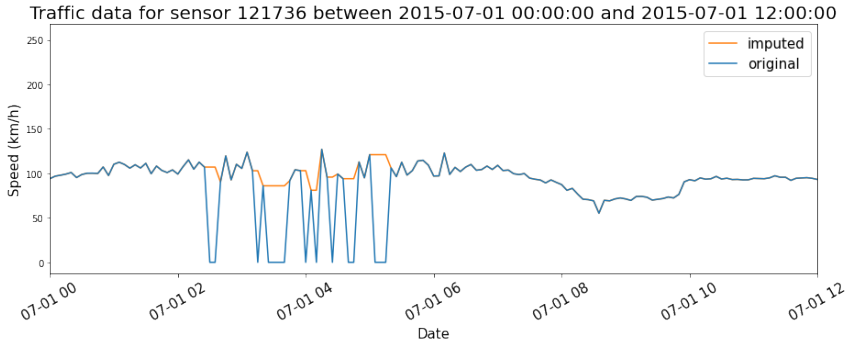
5.5.2.2 Last Observation Carried Forward and Next Observation Carried Backward

Last Observation Carried Forward (LOCF) and Next Observation Carried Backward (NOCB) are two distinct imputation techniques used in time series data. LOCF uses the last observed value to fill in missing values that occur after it and NOCB uses the next observations to fill in missing values that occur before it. In this case, they were used together, LOCF before NOCB to fill the missing values since there can be values missing at the beginning and end of the dataset that would be left missing if only one of the techniques were used. The choice of performing LOCF before NOCB comes from past values having influence over future values, so there was the desire to keep that relationship whenever possible. The plotted values of this imputation in a sample of data from

the VCI dataset can be seen in Fig.5.10a for a block of missing values and Fig.5.10b for sparse missing data.



(a) Block imputation



(b) Sparse imputation

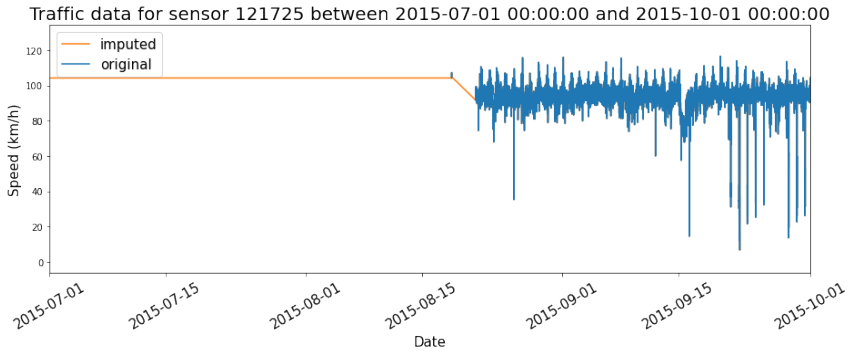
Figure 5.10: Comparison between original data and after LOCF + NOCB imputation in VCI

5.5.2.3 Linear Interpolation

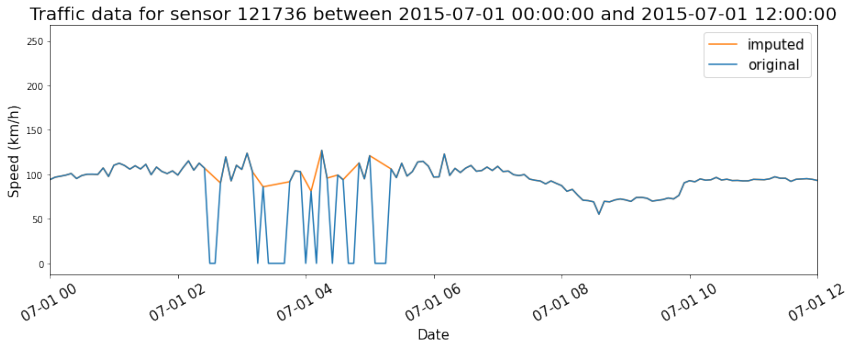
It assumes a linear relationship between the observed values before and after the missing value and estimates the missing value based on this assumption. The plotted values of this imputation in a sample of data from the VCI dataset can be seen in Fig.5.11a for a block of missing values and Fig.5.11b for sparse missing data.

5.5.2.4 MICE - Multiple Imputation with Chained Equations

MICE starts by calculating the column-wise mean for any columns containing missing values and replaces those missing values with the mean. It then proceeds to perform a series of chained regression models to impute each missing value iteratively (Ismiguzel, 2022). Similar to standard regression, MICE utilizes a feature matrix and a target variable for training. The plotted values of this imputation in a sample of data from the VCI dataset can be seen in Fig.5.12a for a block of missing values and Fig.5.12b for sparse missing data.



(a) Block imputation



(b) Sparse imputation

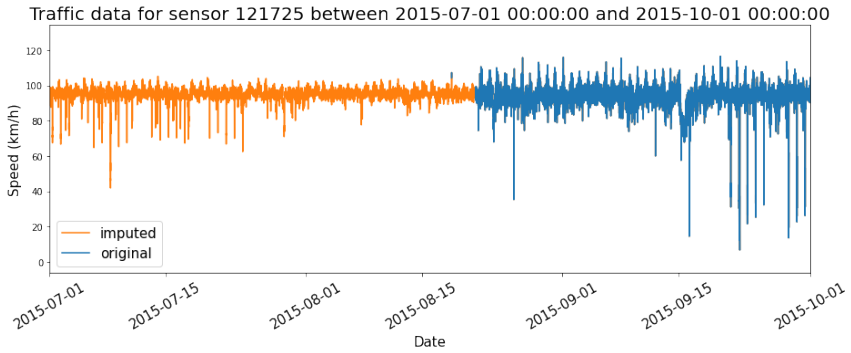
Figure 5.11: Comparison between original data and after interpolation imputation in VCI

5.5.3 Comparing the techniques

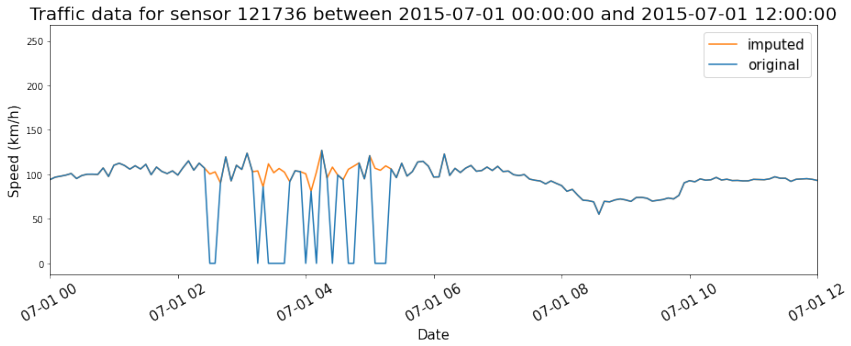
By looking at the different techniques plotted values compared with the rest of the data, it is possible to determine how well they might perform. Still, before using them for the GNN model, it is helpful to get more information on their power to impute values as close to as they would be if they were real. Since for actual missing values, it isn't possible to know what the sensor reading would be it is not possible to see how the techniques perform against each other empirically.

To overcome that, a subset of the VCI dataset was created, containing only sensors with no missing values (which ended up being only four). Then, for that subset, missing values were artificially generated with the following configurations:

- Sparse 4%: Only sparse missing values randomly selected at 4% missing rate for the subset;
- Sparse 8%: Only sparse missing values randomly selected at 8% missing rate for the subset;
- Block I: Only block missing values, randomly selecting two sensors, removing one month's worth of data from one at a random location and from the other removing two weeks also at a random location;
- Block II: Only block missing values, randomly selecting two sensors, removing two month's worth of data from one at a random location and from the other removing one month also at a random location;



(a) Block imputation



(b) Sparse imputation

Figure 5.12: Comparison between original data and after MICE imputation in VCI

- Sparse + Block I: Sparse 4% + Block I used together
- Sparse + Block II: Sparse 4% + Block II used together

Then, the performance of each imputation technique in the different missing data configurations was evaluated using MAE and RMSE metrics. To remove possible bias related to the random choice of the missing data, the metrics were generated by averaging over five runs with different randomly generated missing data. The results can be seen in Table 5.7, and they show that for sparse missing data using interpolation produces the best results, but for blocks of missing data, the MICE approach produces smaller errors. In response to this, a hybrid between these two approaches that uses MICE for blocks and interpolation for sparse missing values was created in it produced the best results for when the both types of missing data exist.

Table 5.7: Empirical evaluation results of the different imputation techniques

	Sparse (4%)		**Sparse (8%)**		**Block I**		**Block II**		**Sparse + Block I**		**Sparse + Block II**	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Mean	0.368	2.739	0.739	3.886	0.307	1.794	0.623	2.643	0.719	3.670	1.300	5.155
LOCF +NOCB	0.154	1,228	0.309	1.728	0.366	2.074	0.786	3.057	0.586	2.903	1.059	3.937
Interpolation	**0.128**	**0.960**	**0.257**	**1.360**	0.297	1.880	0.703	2.957	0.575	2.670	0.961	3.705
MICE	0.275	2.129	0.565	3.060	**0.294**	**1.585**	**0.616**	**2.359**	0.605	3.019	1.210	4.431
MICE+Interpolation	**0.128**	**0.960**	**0.257**	**1.360**	**0.294**	**1.585**	**0.616**	**2.359**	**0.561**	**2.402**	**0.955**	**3.154**

5.5.4 Inclusion in the Data Processing Pipeline

This additional data processing step is added to the existing data processing pipeline described in Section 5.3, before the traffic data is normalized, so the processing steps become as shown in Figure 5.13.

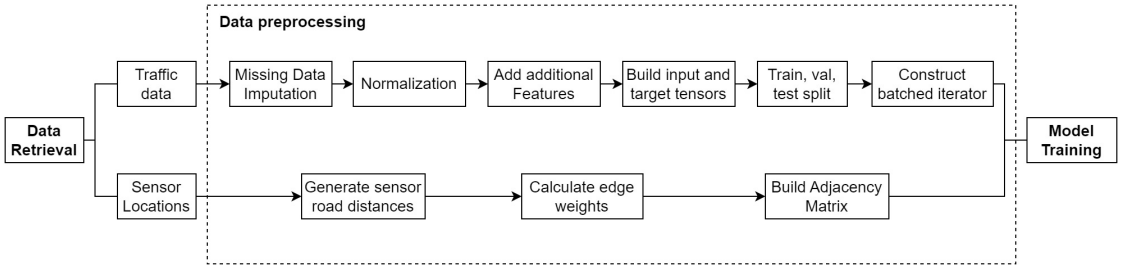


Figure 5.13: Data processing pipeline with missing data imputation

5.6 External Data Incorporation

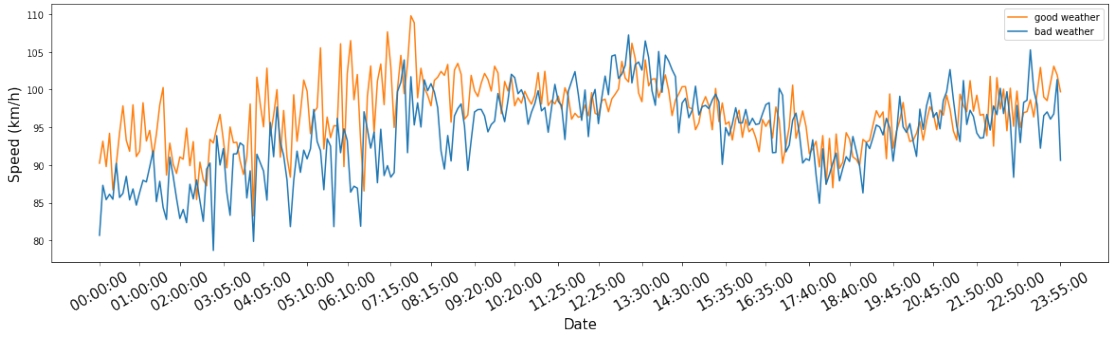
After the base GNN architecture and experimenting with missing data handling techniques, the addition of data from external sources, which were described in Section 5.2, was conducted.

5.6.1 Weather Data

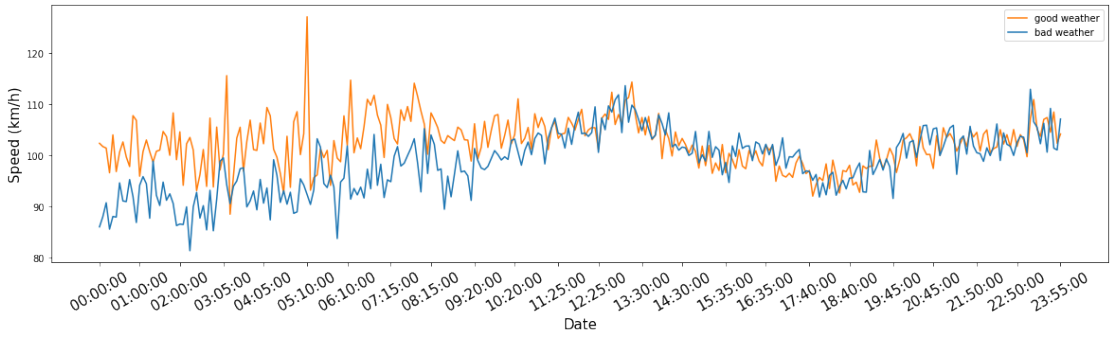
One of the data retrieved to complement the traffic data was the weather conditions in the zone where the road network in question is located, at every timestamp recorded. This choice came from the general notion that on days when there are bad weather conditions, such as heavy rain, low visibility and/or low temperatures there seems to be more congestion in roads. This could have many possible reasons, such as more people driving by car instead of public transportation or walking so as to not be subject to the weather, making the network more congested. This idea is supported by the data used in this work: using VCI as an example, the traffic conditions of 2 days were chosen to be analysed. One day in which the weather conditions were very good (high visibility, no rain, little to no clouds and no significant winds) and one where the weather conditions were very bad (low visibility, heavy rain throughout most part of the day, and significant winds). Both days were the same day of the week (both Sundays) so as to not have weakly patterns affect the analysis as much. The comparison can be seen for three different sensors in Fig.5.14. The chosen sensors are located in distanced places of the network, so as to show that the weather impact is shown across the network.

The weather data was added as an additional feature, as was described for the additional temporal features in Subsection 5.3.3. Several different combinations of such weather condition features were used, which are the following:

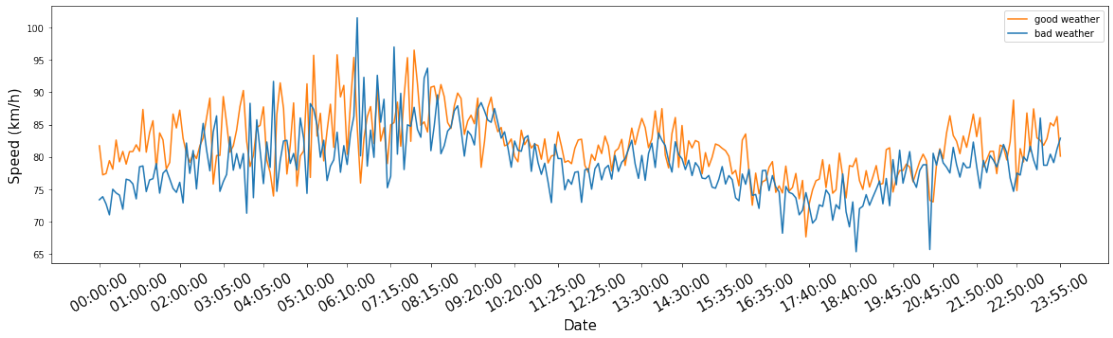
- precip_rate + visibility



(a) Comparison in the traffic of sensor '11'



(b) Comparison in the traffic of sensor '12'



(c) Comparison in the traffic of sensor '23'

Figure 5.14: Comparison between the traffic in a bad weather day and a good weather day

- precip_rate + visibility + wind_spd
- precip_rate + wind_spd
- precip_rate + wind_spd + visibility + temp
- all features: precip_rate + wind_spd + visibility + temp + pod + clouds

This list is not by any means the full exhaustion of all the possible combinations of features, but given the constraint in terms of training time available, it was deemed a reasonable set of possibilities concerning the most promising features. The results of this inclusion can be seen in Section 6.3.

5.6.2 Event Data

The other kind of data retrieved to complement the traffic data was the occurrence of high-attendance events in the area close to the road network in question. This choice came from the fact that such events by having high amounts of people gathering in a specific location at a specific time will directly lead to an influx of transportation needs to that location at that time, changing the usual behaviour of the network's flow.

This effect can be seen in the recorded traffic data, for example, Fig.5.16a shows the evolution of the traffic speed through 24 hours in one of VCI's sensors, represented by number 5 in Fig.5.15 during two different days. One of them, designated 'match day' represents Sunday 20th of September, 2015, in which, according to Futebol Clube do Porto's official agenda[6] there was a match between them and Sport Lisboa e Benfica at 19:15 held at Estádio do Dragão, which is located very close to the VCI road, in particular to the sensor in observation. The other day represented is the Sunday of the week before. In the figure, it is noticeable the decrease in speed observed in the late afternoon, particularly in the times preceding the beginning of the match and then in the evening around the time the match ended, both not present in the 'normal' day. However, this effect is not transversal to the whole network, in Fig.5.16b by plotting the same days but in sensor '25', far away and in the opposite direction of Estádio do Dragão, it is possible to see that the speeds don't differ as they do in sensor '5'. This shows how this interaction between events and traffic is more complex than, for example, the weather since it has different effects in different parts of the network and also at different times.

It is important to note that a single case doesn't prove that all events will significantly impact the network's traffic, but it indicates that it is worth exploring this dependency.

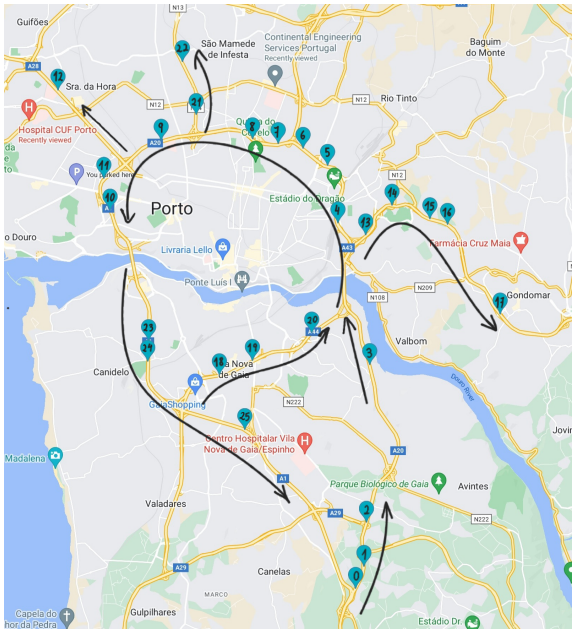
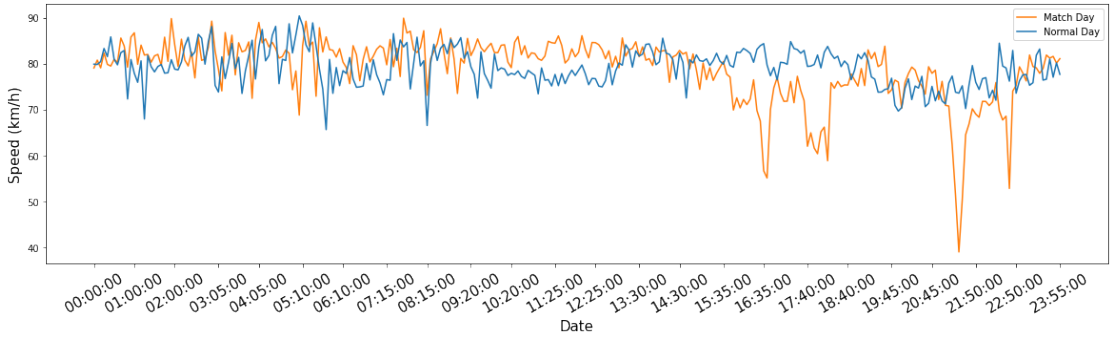
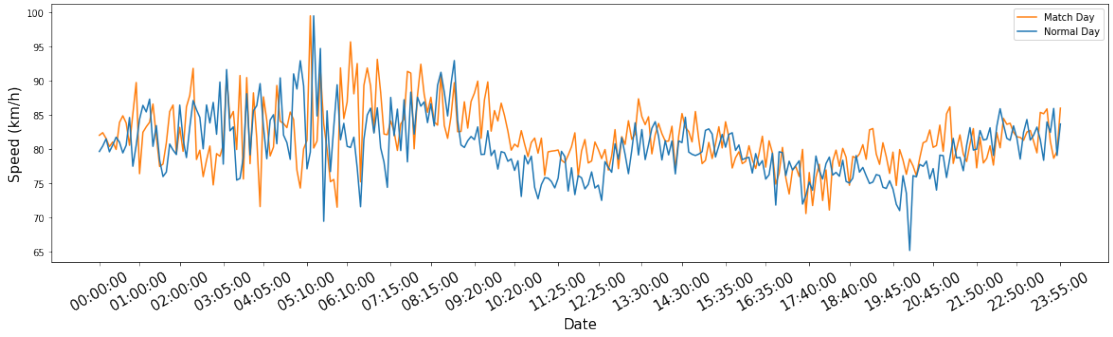


Figure 5.15: VCI Sensor Location and traffic flow

[6] https://www.fcporto.pt/pt/agenda



(a) Comparison in the traffic of sensor ’5’



(b) Comparison in the traffic of sensor ’25’

Figure 5.16: Comparison between the traffic during a match day in Estádio do Dragão and a regular day

Unfortunately, given time restrictions related to the computational time and power required to run the models, it was not possible to carry out this exploration. This means that the PeMS-BAY-2years dataset and the corresponding retrieved event dataset were not used. However, given that the datasets were retrieved and processed, it made sense to keep the mentions to them since they can be used in future research on the topic.

Chapter 6

Empirical Evaluation

In this chapter, the empirical experiences carried out will be shown and discussed. Given their big computational time requirements, different experiments ran on different machines at the same time to economize time. It was relevant to register the training time of the different models, but this became challenging when using different machines for different experiences since they had slightly different performances. The compromise that was deemed adequate for this work was, since there isn't a big discrepancy in the time of different epochs of the same model, to run 10 epochs of each model all on the same machine and register the average of those epochs. During those runs, there was an effort not to have other processes running on the machine so as to have execution time comparisons as reliable as possible. The machines had the same hardware specifications, which were the following: Intel core i7-8700K @3.70GHz processor, 32 GB of RAM and an Nvidia GeForce GTX 1080 graphics card running on Windows 10. The source code was developed in Python 3.10, with PyTorch 2.0, running with CUDA 11.8. Ensuring similar results will only be possible by having these hardware specifications and library versions. To simplify the identification of the different models explored, they were named in this chapter following the components of their network structure, as described in Table 6.1.

Table 6.1: Identification of the names used for the different GNN arquitectures

Name	**Network**
GAT-GRU	GAT as spatial mechanism and GRU as temporal mechanism
GAT-CNN	GAT as spatial mechanism and CNN as temporal mechanism
GAT-GRUAtt	GAT as spatial mechanism and GRU+Attention as temporal mechanism
GAT-CNNAtt	GAT as spatial mechanism and CNN+Attention as temporal mechanism
GCN-GRU	GCN as spatial mechanism and CCN as temporal mechanism
GCN-CNN	GCN as spatial mechanism and CNN as temporal mechanism
GCN-GRUAtt	GCN as spatial mechanism and GRU+Attention as temporal mechanism
GCN-CNNAtt	GCN as spatial mechanism and CNN+Attention as temporal mechanism
Cheb-GRU	ChebConv as spatial mechanism and CCN as temporal mechanism
Cheb-CNN	ChebConv as spatial mechanism and CNN as temporal mechanism
Cheb-GRUAtt	ChebConv as spatial mechanism and GRU+Attention as temporal mechanism
Cheb-CNNAtt	ChebConv as spatial mechanism and CNN+Attention as temporal mechanism

6.1 Base Graph Neural Network Architecture

As explained in the methodology, here, the different GNN Architecture explored were compared in the three datasets, and the two best-performing ones on a base configuration of hyperparameters were chosen to be further explored with different hyperparameter settings.

6.1.1 Model Comparisons in Each Dataset

For this experiment, the settings were the following:

- Learning rate: 0.01
- Number of training epochs: 100
- Optimizer: Adam
- Loss function: Mean Squared Error
- Historical Features: 12
- Prediction Horizons: 12
- Number of Spatio-temporal blocks: 2
- Time features: using only the 'Time in day'

Table 6.2 shows the performance of all the explored models in regard to the PeMS-BAY dataset. It shows the values of MAE, MAPE and RMSE in 3 prediction horizons: 15, 30 and 60 minutes, as well as the average time each epoch takes to train in seconds. It is important to note that the models generate predictions for 12 horizons, but only these three are shown since having them all displayed would make the results very dense and hard to understand. This principle will be followed throughout the tables of this Chapter.

Table 6.2: Comparison of the different models on the PeMS-BAY dataset

Models	15 min			30 min			60 min			Training time per epoch (s)
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	
GAT-GRU	3.706	8.009	6.491	4.551	9.921	6.957	5.348	10.672	7.848	108.345
GAT-CNN	3.763	8.015	6.471	4.698	9.915	6.955	5.245	10.933	7.756	116.324
GAT-GRUAtt	3.579	8.217	6.167	3.869	9.215	6.500	4.647	10.755	7.410	111.876
GAT-CNNAtt	3.409	7.988	6.064	3.799	8.904	6.409	4.543	10.543	7.220	120.320
GCN-GRU	4.359	9.002	6.500	4.892	10.543	8.205	5.994	11.418	8.894	101.456
GCN-CNN	4.466	9.107	6.545	4.952	10.567	8.189	6.082	11.608	8.955	107.980
GCN-GRUAtt	3.722	8.126	6.596	4.619	9.978	7.142	5.411	10.890	7.912	105.876
GCN-CNNAtt	3.804	8.199	6.708	4.685	10.001	7.150	5.587	11.103	7.992	110.435
Cheb-GRU	3.724	8.107	6.455	4.599	9.901	6.944	5.407	10.683	7.798	129.435
Cheb-CNN	3.748	8.122	6.399	4.651	9.915	7.074	5.290	10.901	7.795	121.003
Cheb-GRUAtt	**1.869**	**3.920**	**3.341**	**2.406**	**5.428**	**4.495**	**3.077**	**7.463**	**5.736**	131.120
Cheb-CNNAtt	3.420	8.057	6.103	3.886	8.998	6.497	4.612	10.602	7.295	123.327

One thing that all models have in common results-wise is that the predictions get progressively worse for bigger horizons, which is in line with what is expected since the further away in time we are, the more difficult it is to make good predictions.

Comparing the results of the same temporal mechanism used with different spatial mechanisms, it is noticeable that GCN produces the worst results out of the three mechanisms, which is in line with the fact that this technique is less advanced and was then improved with Chebyshev polynomials to become GhebConv. Comparing ChebConv with GAT is not as clear. The results show similar performances across almost all temporal mechanisms. There is the exception of Cheb-GRUAtt, which significantly outperforms the other ChebConv-based models.

In terms of comparing the results of the same spatial mechanism used with different temporal mechanisms, the temporal mechanisms that have temporal attention (GRUAtt and CNNatt) significantly outperform the ones without attention, so it can be said that having the attention helps to extract the most relevant time dependencies. In terms of choosing between GRUAtt and CNNAtt, GRUAtt tends to have better results but not by a big difference, with the exception of GAT-CNNAtt, that is better than GAT-GRUAtt.

Now looking into other datasets, Table 6.3 shows the performance of all the explored models in the METR-LA dataset.

Table 6.3: Comparison of the different models on the METR-LA dataset

Models	15 min			30 min			60 min			Avg. time
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	p/ Epoch (s)
GAT-GRU	5.420	15.704	8.524	5.755	17.079	9.108	6.265	19.633	10.111	53.498
GAT-CNN	5.446	15.728	8.567	5.806	17.112	9.143	6.234	19.678	10.099	50.349
GAT-GRUAtt	4.978	15.626	8.851	5.342	16.648	9.219	5.899	18.606	9.974	58.457
GAT-CNNAtt	4.877	15.501	8.713	5.190	16.547	9.157	5.785	18.457	9.869	54.630
GCN-GRU	5.890	15.954	8.785	6.102	17.789	9.324	6.654	19.996	10.302	48.768
GCN-CNN	5.921	16.023	8.798	6.132	17.819	9.335	6.678	19.952	10.343	46.674
GCN-GRUAtt	5.551	15.799	8.620	5.865	17.240	9.193	6.370	19.769	10.206	51.348
GCN-CNNAtt	5.557	15.823	8.652	5.898	17.255	9.204	6.362	19.787	10.199	49.540
Cheb-GRU	5.472	15.734	8.562	5.769	17.102	9.112	6.276	19.648	10.102	81.234
Cheb-CNN	5.493	15.789	8.581	5.797	17.123	9.145	6.291	19.667	10.109	78.965
Cheb-GRUAtt	**2.995**	**8.201**	**5.621**	**3.622**	**10.748**	**6.876**	**4.586**	**14.770**	**8.456**	83.439
Cheb-CNNAtt	4.983	15.637	8.865	5.358	16.632	9.207	5.909	18.651	9.921	80.735

Again, the predictions get progressively worse for bigger horizons, as in PeMS-BAY. The performance of the models in METR-LA is worse than in PeMS-BAY, in general across all models. The performance of the temporal and spatial mechanisms follows the tendencies seen in PeMS-BAY.

Table 6.4 shows the performance of all the explored models in the VCI dataset.

Again, the performance of the different mechanisms follows the trend of the previous two datasets.

Comparing the results across the three datasets, some conclusions can be extracted:

- The different models show the same relative performance against each other (i.e. if a model performs better than some other model in one of the datasets it also performs better in the

Table 6.4: Comparison of the different models on the VCI dataset

Models	15 min			30 min			60 min			Avg. time
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	p/ Epoch (s)
GAT-GRU	5.129	9.318	8.458	5.8298	10.612	9.001	5.778	11.213	9.907	72.789
GAT-CNN	5.138	9.337	8.476	5.8265	10.643	9.024	5.785	11.227	9.9019	64.120
GAT-GRUAtt	4.620	8.615	7.745	5.123	10.612	8.798	5.643	11.110	9.899	75.675
GAT-CNNAtt	4.418	7.642	7.353	4.807	8.692	8.265	5.412	10.254	9.495	67.212
GCN-GRU	5.965	10.132	8.945	6.043	11.537	9.788	6.655	11.997	10.337	60.245
GCN-CNN	5.973	10.097	8.959	6.051	11.548	9.736	6.678	12.006	10.341	57.127
GCN-GRUAtt	5.551	9.799	8.620	5.865	11.240	9.224	6.437	11.769	10.064	62.900
GCN-CNNAtt	5.512	9.650	8.589	5.920	11.320	9.193	6.370	11.710	10.178	59.976
Cheb-GRU	4.993	9.255	8.487	5.103	10.855	8.555	5.625	11.899	9.737	119.765
Cheb-CNN	5.009	9.267	8.496	5.148	10.855	8.543	5.678	11.850	9.702	116.953
Cheb-GRUAtt	**4.017**	**6.750**	**6.677**	**4.601**	**8.453**	**7.892**	**5.451**	**10.668**	**9.450**	122.710
Cheb-CNNAtt	4.345	6.925	6.754	4.601	8.591	8.214	5.667	10.826	9.633	118.678

other datasets) on the different datasets, which makes it possible to consider that they are capable of being applied to different use cases.

- The results are overall better in PeMS-BAY, followed by VCI and then METR-LA. This could have to do with the ratios of missing value occurrence in each dataset, given that PeMS-BAY has a very small amount of missing values, then VCI sits in the middle, and METR-LA has the most.

- In terms of training time, the models have the same relative performance across all datasets, but the same model in different datasets takes different times. This makes sense since the datasets have different dimensions, both in sensor number and duration of the recorded period. Models take the most time to run on PeMS-BAY, followed by VCI and then METR-LA.

- Models using GRU as the temporal mechanism tend to be slower than the ones using CNN, which goes in line with the theory behind the two architectures.

- The temporal attention mechanism also minimally slows down training.

- Models with ChebConv as their spatial mechanism are slower than those with GAT or GCN. PeMS-BAY is the largest dataset in the number of sensors (it has 325) and has the same recorded period as VCI (6 months), but VCI has only 26 sensors. METR-LA has 206 sensors and 4 months of recorded period. This gives the indication that the recorded period length has more impact than the number of sensors. Otherwise, the models would run faster in VCI than in METR-LA.

From now on, as they showed better performance in all datasets and for all prediction times, the Cheb-GRUAtt and GAT-CNNAtt models will be the ones in which the following experiments will be performed.

6.1.2 Effect of Hyperparameters

After the top-performing models with the initial hyperparameters configuration were discovered, being Cheb-GRUAtt and GAT-CNNAtt, these models were tested with different hyperparameter configurations to try to improve their performance further.

6.1.2.1 Effect of the Optimizer

For this experiment, the settings, apart from the optimizer, were the following:

- Learning rate: 0.01
- Number of training epochs: 100
- Loss function: Mean Squared Error
- Historical Features: 12
- Prediction Horizons: 12
- Number of Spatio-temporal blocks: 2
- Time features: using only the 'Time in day'

Tables 6.5, 6.6, and 6.7 contain the results of the different optimizers in the PeMS-BAY, METR-LA and VCI datasets, respectively, for the Cheb-GRUAtt model. In regards to this model, in METR-LA, the optimizer with the best results is Adam across all metrics and prediction horizons. In PeMS-BAY, this only verifies on 30 and 60-minute horizons, and for 15 Adagrad gets better results. In the case of VCI, the results are not as clear: for a 15-minute horizon, Adagrad is better, but for 30 and 60-minute horizons the results are very similar across all optimizers.

In terms of the time needed for training, the difference between the optimizers is minimal.

Given that, for two of the datasets (METR-LA and PeMS-BAY), Adam is better, and in VCI the difference isn't very significant, the following experiences with this model will be performed using the Adam optimizer.

Table 6.5: Comparison of the different optimizers used with the Cheb-GRUAtt on the PeMS-BAY dataset

Optimizers	15 min			30 min			60 min			Avg. time
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	p/ Epoch (s)
Adam	1.869	3.920	**3.341**	**2.406**	**5.428**	**4.495**	**3.077**	**7.463**	**5.736**	131.120
AdamW	1.887	3.886	3.383	2.428	5.390	4.527	3.186	7.640	5.785	132.450
Adagrad	**1.714**	**3.835**	3.353	2.306	5.482	4.605	3.143	7.960	6.056	132.970

Now, for the GAT-CNNAtt model, Tables 6.8, 6.9, and 6.10 contain the results of the different optimizers in the PeMS-BAY, METR-LA and VCI datasets, respectively. Here the predominance of better results with Adam that was seen on the previous model does not verify. In METR-LA

Table 6.6: Comparison of the different optimizers used with the Cheb-GRUAtt on the METR-LA dataset

Optimizers	15 min			30 min			60 min			Avg. time
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	p/ Epoch (s)
Adam	**2.995**	**8.201**	**5.621**	**3.622**	**10.748**	**6.876**	**4.586**	**14.770**	**8.456**	83.439
AdamW	4.390	17.542	8.373	4.743	18.421	8.880	5.325	19.900	9.697	83.248
Adagrad	4.442	17.670	8.489	4.848	18.603	8.905	5.554	20.242	9.707	83.952

Table 6.7: Comparison of the different optimizers used with the Cheb-GRUAtt on the VCI dataset

Optimizers	15 min			30 min			60 min			Avg. time
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	p/ Epoch (s)
Adam	4.017	6.750	6.677	4.601	8.453	7.892	5.451	10.668	9.450	122.710
AdamW	3.954	6.905	6.653	4.545	8.565	**7.853**	5.402	**10.663**	**9.229**	121.437
Adagrad	**3.764**	**6.578**	**6.605**	**4.397**	**8.433**	7.906	**5.270**	10.783	9.476	124.789

Table 6.8: Comparison of the different optimizers used with the GAT-CNNAtt on the PeMS-BAY dataset

Optimizers	15 min			30 min			60 min			Avg. time
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	p/ Epoch (s)
Adam	3.409	7.988	6.064	3.799	8.904	6.409	4.543	10.543	7.220	120.32
AdamW	3.089	6.734	5.816	3.199	6.994	5.925	3.340	7.513	6.151	120.45
Adagrad	**2.715**	**6.293**	**5.085**	**2.834**	**6.531**	**5.246**	**3.1525**	**7.276**	**5.735**	120.81

Table 6.9: Comparison of the different optimizers used with the GAT-CNNAtt on the METR-LA dataset

Optimizers	15 min			30 min			60 min			Avg. time
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	p/ Epoch (s)
Adam	4.877	15.501	8.713	5.190	16.547	9.157	5.785	18.457	9.869	54.630
AdamW	4.882	15.929	8.657	5.057	16.497	9.059	5.459	17.808	9.883	56.604
Adagrad	**3.840**	**11.725**	**6.957**	**4.606**	**12.764**	**7.496**	**4.670**	**14.983**	**8.477**	55.490

Table 6.10: Comparison of the different optimizers used with the GAT-CNNAtt on the VCI dataset

Optimizers	15 min			30 min			60 min			Avg. time
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	p/ Epoch (s)
Adam	**4.418**	**7.642**	**7.019**	**4.807**	**8.365**	**8.265**	**5.413**	**10.254**	9.495	67.212
AdamW	4.501	7.738	7.313	4.907	8.796	8.318	5.545	10.435	9.325	63.962
Adagrad	4.610	8.346	7.750	4.947	9.141	8.269	5.552	10.586	**9.201**	64.365

and PeMS-BAY, Adagrad is better across all metrics and horizons. This does not hold for VCI, where Adam performs better.

Comparing the results of the two models, it is not possible to conclude that one particular optimizer is better than the other for this problem, given that the two models work better with different optimizers. One conclusion that is possible to extract is that the optimizers don't have a significant influence on the training time. It is also possible to observe that the results in VCI tend to work differently than for the other datasets. This dataset is a bit more distinct from the others, having a smaller network of sensors that are spaced more apart, which could explain the different

behaviour, but that isn't enough information to conclude that.

6.1.2.2 Effect of the Learning Rate

For this experiment, the settings, apart from the learning rate, were the following:

- Optimizer: Adam (for the Cheb-GRUAtt model) / Adagrad (for the GAT-CNNAtt model)
- Loss function: Mean Squared Error
- Historical Features: 12
- Prediction Horizons: 12
- Number of Spatio-temporal blocks: 2
- Time features: using only the 'Time in day'

The goal here was to test learning rates smaller and bigger than 0.01 (the value used on the previous tests). Given the expected behaviour of experimenting with learning rates, starting with a bigger learning rate (that moves too fast and skips the optimal weights), the results would get progressively better until a 'sweet spot' is reached, where the optimal weights can be reached within the given epochs, and after that, the results would get worse again, since more epochs would be needed to converge. Following this principle, the procedure followed was to try progressively lower learning rates until the results started deteriorating.

Table 6.11 has the results of training the Cheb-GRUAtt model with different learning rates over 100 epochs on PeMS-BAY. It is possible to see that the learning rates 0.001 and 0.0005 have the best results, with 0.001 beating 0.0005 in the 15-minute horizon but 0.0005 is better for the bigger horizons.

Table 6.11: Results of different learning rates with 100 epochs for Cheb-GRUAtt on PeMS-BAY

Learning Rate	Nr of Epochs	15 min			30 min			60 min			Avg. time p/ Epoch (s)
		MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	
0.020	100	1.842	3.963	3.403	2.502	5.643	4.630	3.268	7.915	5.972	131.674
0.015		1.825	3.964	3.373	2.392	5.490	4.545	3.195	7.698	5.857	130.322
0.010		1.869	3.920	3.341	2.406	5.428	4.495	3.077	7.463	5.736	131.120
0.005		1.789	3.793	3.295	2.345	5.369	4.467	3.069	7.420	5.687	132.002
0.0025		1.709	3.695	3.248	2.274	5.286	4.420	3.004	7.328	5.638	131.107
0.001		**1.673**	**3.659**	**3.236**	2.275	5.256	**4.419**	3.029	7.319	5.660	130.992
0.0005		1.678	3.673	3.241	**2.266**	**5.231**	4.425	**2.979**	**7.214**	**5.647**	130.717
0.00025		1.687	3.667	3.247	2.270	5.242	4.434	2.985	7.285	5.669	131.203

For METR-LA, the results are in Table 6.12 and don't show such a clear picture, with 0.005 being better.

Table 6.13 contains the results in the VCI dataset, where it is possible to see that the best learning rate across all horizons and metrics was the smallest one tested, 0.00025, with results getting better and better the lower the learning rate is. This could indicate that an even smaller

Table 6.12: Results of different learning rates with 100 epochs for Cheb-GRUAtt on METR-LA

Learning Rate	Nr of Epochs	15 min			30 min			60 min			Avg. time p/ Epoch (s)
		MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	
0.020	100	2.908	7.907	5.480	3.548	10.456	6.721	4.483	14.190	8.241	83.135
0.015		2.916	7.911	6.732	3.547	10.396	6.732	4.467	14.112	8.248	82.854
0.010		2.995	8.201	5.621	3.622	10.748	6.876	4.586	14.770	8.456	82.450
0.005		3.139	7.907	5.493	**3.532**	10.350	**6.728**	**4.408**	**14.028**	8.261	83.123
0.0025		2.900	7.922	**5.001**	3.533	10.438	6.750	4.477	14.206	8.258	83.637
0.001		2.902	7.944	5.496	3.544	10.434	6.740	4.493	14.136	**8.237**	82.500
0.0005		**2.877**	7.832	5.500	3.535	**10.349**	6.748	4.523	14.159	8.266	83.439
0.00025		2.895	**7.825**	5.508	3.593	10.457	6.781	4.646	14.504	8.334	83.127

value could still provide better results, but since in the other datasets, this small learning rate got worse results than some bigger ones, the tests stopped at 0.00025.

By observing the results on the three datasets, it is possible to say that they follow the same tendencies but not exactly the same results, not being able to pinpoint one learning rate that systematically is better in all three datasets. Since the goal of this work is to try to generalize the process to fit different datasets, it was decided that 0.0005, which was the best performing on PeMS-BAY but also worked well with the two others, would now be used in all the remaining tests.

Table 6.13: Results of different learning rates with 100 epochs for Cheb-GRUAtt on VCI

Learning Rate	Nr of Epochs	15 min			30 min			60 min			Avg. time p/ Epoch (s)
		MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	
0.020	100	4.094	6.968	6.842	4.687	8.773	8.091	5.572	11.215	9.696	120.065
0.015		4.107	7.174	6.853	4.686	8.870	8.082	5.534	11.062	9.624	119.645
0.010		4.017	6.750	6.677	4.601	8.453	7.892	5.451	10.668	9.450	122.710
0.005		3.893	6.560	6.574	4.471	8.227	7.754	5.309	10.439	9.223	120.602
0.0025		3.845	6.399	6.493	4.452	8.191	7.703	5.274	10.432	9.167	120.749
0.001		3.835	6.554	6.493	4.425	8.237	7.697	5.229	10.353	9.148	121.689
0.0005		3.800	6.452	6.458	4.397	8.180	7.658	5.210	10.399	9.151	120.941
0.00025		**3.791**	**6.394**	**6.445**	**4.375**	**8.076**	**7.621**	**5.181**	**10.285**	**9.077**	118.870

Now concerning the other model, GAT-CNNAtt, the result in PeMS-BAY, METR-LA and VCI datasets can be seen, respectively, in Tables 6.14, 6.15 and 6.16. For PeMS-BAY, the best learning rate of the ones tested is 0.05, for METR-LA it is 0.015 and for VCI the results are not as clear between 0.020 and 0.015. As it happenned with the other model, different datasets have better results with different learning rate, but 0.015 was the one chosen to continue the next tests in this model.

In terms of the effect the learning rate has on the training time, it does not give a noticeable influence.

6.1.2.3 Effect of the additional time features

For this experiment, the settings, apart from the learning rate, were the following:

- Number of training epochs: 100

Table 6.14: Results of different learning rates and with 100 epochs for GAT-CNNAtt on PeMS-BAY

Learning Rate	Nr of Epochs	15 min			30 min			60 min			Avg. time p/ Epoch (s)
		MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	
0.020		2.642	6.364	5.071	2.741	6.654	5.247	3.094	7.173	5.620	121.193
0.015		2.751	6.567	5.211	2.849	6.766	5.359	3.117	7.393	5.754	121.505
0.010	100	2.715	6.293	5.085	2.834	6.531	5.246	3.1525	7.276	5.735	120.810
0.005		**2.602**	**6.172**	**4.933**	**2.727**	**6.447**	**5.123**	**3.009**	**7.170**	**5.594**	121.225
0.0025		2.921	6.803	5.256	3.020	7.041	5.401	3.327	7.797	5.841	121.505

Table 6.15: Results of different learning rates and with 100 epochs for GAT-CNNAtt on METR-LA

Learning Rate	Nr of Epochs	15 min			30 min			60 min			Avg. time p/ Epoch (s)
		MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	
0.020		4.150	13.000	7.582	4.337	13.789	7.950	4.762	15.578	8.691	54.899
0.015		**3.683**	**10.872**	**6.687**	**3.968**	**12.068**	**7.301**	**4.530**	**14.435**	**8.344**	55.184
0.010	100	3.840	11.725	6.957	4.606	12.764	7.496	4.670	14.983	8.477	55.490
0.005		4.318	13.233	7.733	4.572	14.380	8.281	5.063	16.213	9.130	55.062
0.0025		4.541	14.630	8.012	4.781	15.567	8.421	5.193	17.069	9.088	54.851

Table 6.16: Results of different learning rates with 100 epochs for GAT-CNNAtt on VCI

Learning Rate	Nr of Epochs	15 min			30 min			60 min			Avg. time p/ Epoch (s)
		MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	
0.020		**4.444**	**7.572**	**7.177**	4.848	**8.696**	**7.949**	**5.426**	10.512	10.482	66.538
0.015		**4.444**	7.802	7.292	**4.800**	8.813	8.006	5.437	**10.461**	**9.176**	66.919
0.010	100	4.610	8.346	7.750	4.947	9.141	8.269	5.552	10.586	9.201	64.365
0.005		5.075	9.597	8.635	5.435	10.416	9.146	5.965	11.738	9.986	66.625
0.0025		5.211	10.374	9.103	5.536	11.078	9.585	6.013	12.124	10.303	67.116

- Optimizer: Adam (for the Cheb-GRUAtt model) / Adagrad (for the GAT-CNNAtt model)
- Learning rate: 0.0005 (for the Cheb-GRUAtt model) / 0.015 (for the GAT-CNNAtt model)
- Loss function: Mean Squared Error
- Historical Features: 12
- Prediction Horizons: 12
- Number of Spatio-temporal blocks: 2

To make the tables more compact, abbreviations were used to represent the different combinations of weather features, presented in Table 6.17.

The goal was to see if adding additional time features would benefit the results. In theory, this could help the model to find patterns in the temporal aspect of data, i.e. between the same days of the week, distinguish between weekdays and weekends and also between different times of the day. it is important to note that the problem addressed is a short-term prediction, in which some of these patterns might not have as much influence, so the results should help to answer these questions.

Table 6.17: Abbreviations for time features to use in the experiments' tables

Time Features Used	**Abbreviations**
No time features	None
Time in day	Day
Time in Day + Day in Week	Day + Week
Time in Day + Is weekend	Day + Weekend
Time in Day + Day in Week + Hour in Day	Day + Week + Hour
All time features	All

Tables 6.18, 6.19 and 6.20 show the results of adding different combinations of time features to the input data of the Cheb-GRUAtt model in PeMS-BAY, METR-LA and VCI datasets, respectively.

Table 6.18: Results of Cheb-GRUAtt with the use of different combinations of time features in PeMS-BAY

Time Features	15 min			30 min			60 min			Avg. time
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	p/ Epoch (s)
None	1.708	3.696	3.407	2.419	5.640	4.866	3.456	8.687	6.615	131.337
Day	**1.678**	3.673	3.241	2.266	5.231	4.425	2.979	7.214	5.647	130.717
Day + Week	1.688	3.705	**3.230**	**2.233**	5.207	**4.375**	**2.909**	7.109	**5.533**	132.326
Day + Weekend	1.692	**3.651**	3.234	2.255	**5.190**	4.389	2.931	**7.086**	5.568	132.010
Day + Week + Hour	1.696	3.724	3.240	2.255	5.252	4.400	2.936	7.159	5.567	133.132
All	1.692	3.679	3.233	2.667	5.230	4.388	2.946	7.119	5.558	130.956

Table 6.19: Results of Cheb-GRUAtt with the use of different combinations of time features in METR-LA

Time Features	15 min			30 min			60 min			Avg. time
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	p/ Epoch (s)
None	**2.850**	**7.720**	5.565	3.554	10.477	6.924	4.695	14.867	8.666	81.829
Day	2.877	7.832	5.500	**3.535**	**10.349**	6.748	**4.523**	14.159	8.266	83.439
Day + Week	2.960	7.980	**5.466**	3.651	10.519	**6.692**	4.657	14.239	8.173	83.218
Day + Weekend	2.928	7.944	5.477	3.600	10.400	**6.692**	4.592	**14.029**	**8.149**	83.376
Day + Week + Hour	2.977	7.953	5.475	3.678	10.478	6.672	4.698	14.205	8.163	83.556
All	2.976	8.084	5.477	3.646	10.519	6.679	4.673	14.174	8.151	83.015

Table 6.20: Results of Cheb-GRUAtt with the use of different combinations of time features in VCI

Time Features	15 min			30 min			60 min			Avg. time
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	p/ Epoch (s)
None	**3.785**	**6.432**	6.594	4.451	8.378	7.928	5.382	10.963	9.595	123.141
Day	3.800	6.452	**6.458**	**4.397**	**8.180**	**7.658**	**5.210**	**10.399**	**9.151**	120.941
Day + Week	3.857	6.540	6.525	4.468	8.307	7.771	5.268	10.510	9.302	119.704
Day + Weekend	3.931	6.584	6.598	4.520	8.266	7.777	5.316	10.409	9.230	119.653
Day + Week + Hour	3.836	6.453	6.487	4.464	8.204	7.724	5.324	10.438	9.259	120.032
All	3.963	6.600	6.568	4.571	8.238	7.780	5.397	10.497	9.329	120.362

In PeMS-BAY, it is possible to see that having no time features produces the worst results, but in terms of the best results, it is difficult to see which combination is best. 'Time in Day + Day in Week' is better in most metrics, but 'Time in Day' and 'Time in Day + Is Weekend' perform

better in others. In METR-LA, having no time features performs better for the smallest prediction horizon, but then it starts getting worse than other alternatives. 'Time in Day' is better in some cases, but in others, it is 'Time in Day + Is Weekend'. In the case of VCI, having no time features works better in the smallest prediction horizon, but in the two other horizons, 'Time in Day' is better.

It is worth noting that the differences between the results are very small, with differences sometimes only on the second and third decimal cases, giving them little significance.

The choice here was to continue using 'Time in Day' as the only time feature with this model.

Tables 6.21, 6.22 and 6.23 show the results of adding different combinations of time features to the input data of GAT-CNNAtt in PeMS-BAY, METR-LA and VCI datasets, respectively.

Table 6.21: Results of GAT-CNNAtt with the use of different combinations of time features in PeMS-BAY

Time Features	15 min			30 min			60 min			Avg. time
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	p/ Epoch (s)
None	2.769	**6.303**	5.281	2.901	**6.581**	5.497	3.281	7.463	6.035	120.005
Day	2.751	6.567	5.211	2.849	6.766	5.359	3.117	7.393	5.754	121.505
Day + Week	**2.687**	6.463	5.185	**2.805**	6.813	5.390	3.089	7.645	5.893	123.764
Day + Weekend	2.707	6.377	**5.119**	2.812	6.642	**5.277**	**3.081**	**7.326**	**5.694**	123.445
Day + Week + Hour	2.746	6.561	5.208	2.878	6.881	5.426	3.186	7.609	5.883	124.842
All	2.755	6.359	5.226	2.852	6.864	5.472	3.108	7.615	5.818	126.565

Table 6.22: Results of GAT-CNNAtt with the use of different combinations of time features in METR-LA

Time Features	15 min			30 min			60 min			Avg. time
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	p/ Epoch (s)
None	4.979	16.239	9.360	5.387	18.062	9.955	6.021	20.916	10.824	53.081
Day	**3.683**	**10.872**	**6.687**	**3.968**	**12.068**	**7.301**	**4.530**	**14.435**	**8.344**	55.184
Day + Week	5.357	15.962	8.733	5.618	16.997	9.135	6.081	18.905	9.808	55.706
Day + Weekend	4.541	13.532	7.989	4.738	14.250	8.354	5.267	15.711	8.985	55.367
Day + Week + Hour	4.594	14.054	7.870	4.982	15.323	8.416	5.644	17.504	8.416	56.513
All	4.324	12.547	7.497	4.603	13.775	8.072	5.137	16.038	9.035	57.432

Table 6.23: Results of GAT-CNNAtt with the use of different combinations of time features in VCI

Time Features	15 min			30 min			60 min			Avg. time
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	p/ Epoch (s)
None	**4.220**	**7.670**	**7.174**	**4.625**	8.873	8.057	**5.351**	10.776	9.421	70-147
Day	4.444	7.802	7.292	4.800	**8.813**	**8.006**	5.437	**10.461**	**9.176**	66.919
Day + Week	4.855	8.938	8.068	5.261	9.085	8.867	5.998	11.137	10.088	64.421
Day + Weekend	4.463	8.061	7.553	4.844	8.988	8.212	5.526	10.606	9.458	67.075
Day + Week + Hour	5.650	10.523	9.301	5.896	11.072	9.776	6.362	11.974	10.478	63.894
All	5.161	9.197	9.055	5.389	9.867	9.509	5.744	10.936	10.390	65.300

The influence the time features have when used with the three datasets is different. In the case of PeMS-BAY the results are all very similar, but for the bigger horizons, the 'Time in Day + Day in Week' combination produces better results. In the case of METR-LA, the results show

more difference between the different combinations, and using only 'Time in Day' produces the best results. For VCI, the difference between the different combinations is less noticeable than in METR-LA but more than in PeMS-BAY. Using no time features works best for the smallest prediction horizon, but for the bigger ones, 'Time in Day' produces better results in terms of MAPE and RMSE.

In terms of training time, the addition or removal of time features does not produce a significant impact.

6.1.2.4 Effect of the number of Historical Features and Prediction Horizons

For this experiment, the settings, apart from the learning rate, were the following:

- Number of training epochs: 100
- Optimizer: Adam (for the Cheb-GRUAtt model) / Adagrad (for the GAT-CNNAtt model)
- Learning rate: 0.0005 (for the Cheb-GRUAtt model) / 0.015 (for the GAT-CNNAtt model)
- Loss function: Mean Squared Error
- Number of Spatio-temporal blocks: 2
- Time features: 'Time in Day'

The goal of this experiment was to see the impact that changing the number of historical features given as input would change the results for each prediction horizon. Moreover, it was also to see how the models perform on longer prediction, i.e predicting 90 and 120 minutes ahead. As explained in Section 5.3.4, having, for example, 'Historic Features' be 6 and 'Prediction Horizon' be 3, means that $6 \times 5 = 30$ minutes before the given timestamp were used as historical features to predict $3 \times 5 = 15$ minutes into the future from the given timestamp. Tables 6.25 and 6.24 contain the results of different combinations of historical features and prediction horizons for Cheb-GRUAtt and GAT-CNNAtt, respectively. The reason this study was only made on one dataset, comes from the fact that training the models especially when there are more historic features is very slow and it requires a lot of computational time to be able to extract the results to fill the tables. METR-LA is the dataset in which the models train faster, so it was chosen as the only one where this would be tried.

Some conclusions can be extracted from observing the tables:

- By comparing the results on the same prediction horizon, increasing the number of historical features makes the results worse. Comparing the results of prediction up to 60 minutes (12 prediction steps), the results when using 12, 18 or 24 historical features get progressively worse results, and this applies to the other prediction horizons.
- By comparing the use of the same number of historical features for different numbers of prediction horizons, the results, in general, get better the more horizons are predicted.

Table 6.24: Results of Cheb-GRUAtt with different numbers of historical features and output predictions on METR-LA dataset

Hist. Feat.	Pred.	15 min			30 min			60 min			90 min			120 min			Time (s)
		MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	
3	3	2.903	7.821	5.491	-	-	-	-	-	-	-	-	-	-	-	-	21.041
6	3	3.573	10.620	6.796	-	-	-	-	-	-	-	-	-	-	-	-	21.011
6	6	2.893	7.795	5.480	3.555	10.490	6.746	-	-	-	-	-	-	-	-	-	41.900
12	3	4.529	14.278	8.257	-	-	-	-	-	-	-	-	-	-	-	-	21.192
12	6	4.126	12.670	7.592	4.563	14.320	8.227	-	-	-	-	-	-	-	-	-	42.133
12	12	2.902	7.944	5.496	3.544	10.434	6.734	4.493	14.136	8.237	-	-	-	-	-	-	83.439
18	3	5.287	17.046	9.192	-	-	-	-	-	-	-	-	-	-	-	-	21.341
18	6	4.900	15.667	8.731	5.225	16.853	9.157	-	-	-	-	-	-	-	-	-	42.169
18	12	4.090	12.578	7.593	4.506	14.178	8.209	5.205	16.911	9.164	-	-	-	-	-	-	83.202
18	18	2.935	7.951	5.583	3.573	10.523	6.860	4.552	14.901	8.581	5.342	17.391	9.304	-	-	-	123.031
24	3	5.934	19.196	9.917	-	-	-	-	-	-	-	-	-	-	-	-	21.774
24	6	5.577	18.028	9.552	5.813	18.938	9.855	-	-	-	-	-	-	-	-	-	42.368
24	12	4.979	15.673	8.769	5.292	16.863	9.173	5.859	18.991	9.847	-	-	-	-	-	-	83.283
24	18	4.143	12.765	7.634	4.567	14.345	8.255	5.267	17.357	9.332	5.817	19.049	9.894	-	-	-	124.000
24	24	2.951	8.058	5.561	3.608	10.523	6.797	4.725	14.687	8.474	5.426	17.367	9.395	5.908	19.206	9.976	174.071

Table 6.25: Results of GAT-CNNAtt with different numbers of historical features and output predictions on METR-LA dataset

Hist. Feat.	Pred.	15 min			30 min			60 min			90 min			120 min			Time (s)
		MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	
3	3	3.497	10.051	6.683	-	-	-	-	-	-	-	-	-	-	-	-	26.187
6	3	3.810	11.996	7.163	-	-	-	-	-	-	-	-	-	-	-	-	35.618
6	6	3.903	11.862	7.055	4.178	12.980	7.672	-	-	-	-	-	-	-	-	-	34.248
12	3	5.327	18.657	9.43	-	-	-	-	-	-	-	-	-	-	-	-	56.874
12	6	4.723	14.691	8.652	4.951	15.500	9.022	-	-	-	-	-	-	-	-	-	55.239
12	12	3.683	10.872	6.687	3.968	12.068	7.301	4.530	14.435	8.344	-	-	-	-	-	-	55.356
18	3	4.753	14.136	8.762	-	-	-	-	-	-	-	-	-	-	-	-	77.584
18	6	4.445	13.903	8.006	4.635	14.629	8.424	-	-	-	-	-	-	-	-	-	75.963
18	12	4.883	15.808	8.858	4.983	16.190	9.067	5.318	17.458	9.612	-	-	-	-	-	-	76.270
18	18	5.646	18.698	9.672	5.737	19.090	9.872	6.077	20.314	10.393	6.357	21.368	10.749	-	-	-	76.004
24	3	6.682	24.200	11.8739	-	-	-	-	-	-	-	-	-	-	-	-	98.448
24	6	5.369	17.501	9.819	5.504	18.026	10.058	-	-	-	-	-	-	-	-	-	94.667
24	12	5.921	20.211	10.755	6.105	20.904	11.080	6.567	22.471	11.797	-	-	-	-	-	-	96.765
24	18	5.523	17.527	9.624	5.621	18.102	9.779	5.863	19.232	10.100	6.140	20.369	6.140	-	-	-	96.172
24	24	6.197	20.945	10.594	6.349	21.673	10.840	6.605	22.949	11.258	6.803	23.976	11.605	6.950	24.777	11.891	96.412

- As seen in the other experiments, when observing a single row (results across different horizons for the same configuration), the results get progressively worse as the horizons get bigger.
- The results obtained with Cheb-GRUAtt are better than GAT-CNNAtt, across all combinations in all metrics.
- When looking at the bigger horizons of 90 and 120 minutes, it is possible to see that the results achieved by the models get very degraded, which shows that their applicability is focused towards short-term prediction, as also pointed out by the recent literature. It is still possible to see, however, that the Cheb-GRUAtt model has better results and shows less degradation than the GAT-CNNAtt model.

6.1.3 Effect of the Length of the Training Set

With the retrieval of the PeMS-BAY-2years dataset, it was possible to test if training on a longer period of recorded data would produce better results. For this, 4 different training were carried out: first using only the 6 most recent months, then the 12 more recent, then 18 and finally the whole 24 months. It's important to note that the test set for evaluating the 4 different sets was kept

the same. Given training time restrictions, only the Cheb-GRUAtt Model was experimented on. The settings for this experiment were the following:

- Number of training epochs: 100
- Optimizer: Adam
- Learning rate: 0.0005
- Loss function: Mean Squared Error
- Number of Spatio-temporal blocks: 2
- Time features: 'Time in Day'
- Historical Features: 12
- Prediction Horizons: 12

Table 6.26: Comparison of the results with different training dataset lengths

Dataset	15 min			30 min			60 min			Avg. time
Length	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	per Epoch (s)
6 months	1.280	2.562	2.603	1.634	3.208	3.432	2.046	4.517	4.202	122.346
12 months	**1.198**	**2.402**	**2.561**	**1.524**	**3.201**	**3.346**	**1.890**	**4.097**	**4.080**	255.478
18 months	1.280	2.556	2.602	1.658	3.473	3.421	2.032	4.453	4.171	370.385
24 months	1.234	2.406	2.563	1.592	3.253	3.375	2.110	4.401	4.164	508.349

Table 6.26 shows the results of the different dataset lengths used for training. The results show that increasing the dataset length to 12 months produces better results. This can come from the fact that training with a full year allows the model to capture the full extent of the seasonal dependencies. When extending the results further to 18 months, go back to being similar to training with 6 months. Training with 24 months is slightly better than with 18 but worse than with 12 months. These results indicate that training with one year of data can help the results, but further increasing the training length is not favourable. When looking at the training times, they show a linear increase in time with the length increase, i.e. training with 12 months takes roughly twice the time as training with 6 months and so on. This lack of better results with the increase in dataset length could come from the fact that the less recent data present in those parts happened in a time that was more affected by COVID, which impacted the typical transportation patterns.

6.2 Handling Missing Data

Now that the experiments with the base GNN approach are tackled, the results when using missing data imputation will be explored. From now on, since Cheb-GRUAtt got the best results in all experiments when compared to GAT-CNNAtt, the experiments will only be performed on Cheb-GRUAtt, given the computational time limitations.

For this experiment, the settings were the following:

- Number of training epochs: 100
- Optimizer: Adam
- Learning rate: 0.0005
- Loss function: Mean Squared Error
- Number of Spatio-temporal blocks: 2
- Time features: 'Time in Day'
- Historical Features: 12
- Prediction Horizons: 12

Table 6.27 shows the results of using different missing data imputation techniques on the PeMS-BAY dataset.

Table 6.27: Comparison of the results with the different missing data imputation techniques in the PeMS-BAY dataset

Techniques	15 min			30 min			60 min			Avg. time
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	per Epoch (s)
No imputation	1.678	3.673	3.241	2.266	5.231	4.425	2.979	7.214	5.647	130.717
Mean	1.678	3.680	3.242	2.273	5.253	4.426	2.998	7.248	5.647	131.921
LOCF+NOCB	1.678	3.680	3.243	2.265	5.262	4.426	2.997	7.249	5.647	131.257
Interpolation	1.678	3.679	3.242	2.273	5.253	4.426	2.997	7.247	5.647	131.678
MICE	1.688	3.705	3.260	2.288	5.289	4.459	3.042	7.404	5.728	132.002
MICE+interpolation	1.678	3.679	3.242	2.273	5.253	4.426	2.997	7.247	5.647	131.976

The results here show very little difference between the different techniques and even worsen them in the case of MICE. Since this particular dataset has very little missing data, the impact they have on the results is minimal, so using the missing data imputation techniques did not make a noticeable different. This is why no results were highlighted in the table.

Table 6.28 shows the results of using different missing data imputation techniques on the METR-LA dataset.

Table 6.28: Comparison of the results with the different missing data imputation techniques in the METR-LA dataset

Techniques	15 min			30 min			60 min			Avg. time
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	per Epoch (s)
No imputation	2.877	7.832	5.500	3.535	10.349	6.748	4.523	14.159	8.266	83.439
Mean	2.925	**7.486**	5.503	3.547	**9.538**	**6.318**	**4.446**	**12.481**	**8.006**	83.196
LOCF+NOCB	2.840	7.799	5.486	3.501	10.312	6.700	4.491	14.122	8.221	83.345
Interpolation	2.862	7.665	**5.417**	**3.504**	10.004	6.626	4.464	13.500	8.115	83.542
MICE	**2.861**	7.598	5.531	3.583	9.674	6.679	4.490	12.672	8.054	83.630
MICE+interpolation	2.862	7.665	**5.417**	3.504	10.004	6.626	4.464	13.500	8.114	83.965

In this dataset, the results show a bit more difference. Using the mean, despite its simplicity and not being the preferred technique for time series data imputation, showed the best results, although closely followed most of the time by MICE.

Table 6.29 shows the results when used with VCI.

Table 6.29: Comparison of the results with the different missing data imputation techniques in the VCI dataset

Techniques	15 min			30 min			60 min			Avg. time
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	per Epoch (s)
No imputation	3.800	6.452	6.458	4.397	8.180	7.658	**5.210**	10.399	9.151	120.941
Mean	**3.778**	**6.338**	6.437	4.386	8.129	7.665	5.225	10.451	9.168	123.976
LOCF+NOCB	3.887	6.395	6.522	4.486	8.200	7.704	5.282	10.393	9.181	122.234
Interpolation	3.857	6.373	6.508	4.452	**8.102**	7.693	5.269	10.385	9.157	120.869
MICE	3.825	6.890	**6.430**	**4.417**	8.174	**7.656**	5.221	**10.367**	**9.119**	121.210
MICE+interpolation	3.856	6.442	6.493	4.450	8.179	7.707	5.260	10.402	9.191	124.639

Here the results of the different techniques show a bit more influence than in PeMS-BAY but less than in METR-LA. This can stem from the fact that PeMS-BAY has the least amount of missing data, followed by VCI and then METR-LA. Having less missing data makes the imputation used to fill them less impactful on the overall results. Given the results, it can be understood why most of the previous works on GNNs for traffic forecasting don't pay a lot of attention to the missing data, given that the models, especially when their occurrence is small, can deal well with their presence. But since the results show that more impact is noticed when the presence of missing data is bigger, meaning that the models are not immune to missing data and can benefit from the use of these techniques. Furthermore, this shows that when dealing with datasets where there are even more missing data than in METR-LA, the usage of an imputation technique can have an even more significant impact and should not be overlooked. One detail worth mentioning is that, when comparing the imputation techniques that performed better pre-process the data for the GNN models with the results shown in Table 5.7, where the power of imputation is evaluated on its own, the results don't match up. That is, the techniques that were better at generating values closer to what they would really be, where not the ones that helped the model achieve the best results. This is seen especially in the mean imputation, which did not perform well on its own as imputation but when used as pre-processing for the GNN models, helped them achieve better results.

6.3 Weather Data Incorporation

For this experiment, the settings were the following:

- Number of training epochs: 100
- Optimizer: Adam
- Learning rate: 0.0005
- Loss function: Mean Squared Error
- Number of Spatio-temporal blocks: 2

- Time features: 'Time in Day'
- Missing Data imputation technique: None

To make the tables more compact, abbreviations were used to represent the different combinations of weather features, presented in Table 6.30

Table 6.30: Abbreviations for weather features to use in the experiments' tables

Weather Features Used	**Abbreviations**
No weather features	None
precip_rate + visibility	p + v
precip_rate + visibility + wind_spd	p + v + w
precip_rate + wind_spd	p + w
precip_rate + wind_spd + visibility + temp	p + w + v + t
All features	All

Tables 6.31, 6.32 and 6.33 show the results obtained when adding the weather features to the input of the Cheb-GRUAtt model in PeMS-BAY, METR-LA and VCI, respectively.

Table 6.31: Results with the use of different combinations of weather features in PeMS-BAY

Weather Features	15 min			30 min			60 min			Avg. time
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	p/ Epoch (s)
None	**1.678**	**3.673**	**3.241**	**2.266**	**5.231**	**4.425**	**2.979**	**7.214**	**5.647**	130.717
p + v	1.691	3.746	3.256	2.277	5.320	4.438	2.986	7.314	5.676	132.992
p + v + w	1.781	3.984	3.530	2.412	5.767	4.885	3.396	8.647	6.548	132.659
p + w	1.696	3.726	3.247	2.285	5.294	4.431	2.989	7.270	5.673	132.973
p + w + v + t	1.736	3.899	3.304	2.343	5.482	4.499	3.093	7.501	5.761	132.638
All	1.749	3.824	3.280	2.349	5.389	4.472	3.090	7.393	5.744	134.078

Table 6.32: Results with the use of different combinations of weather features in METR-LA

Weather Features	15 min			30 min			60 min			Avg. time
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	p/ Epoch (s)
None	**2.877**	**7.832**	5.500	**3.535**	**10.349**	**6.748**	**4.523**	**14.159**	**8.266**	83.439
p + v	2.893	7.882	5.517	3.570	10.451	6.781	4.593	14.306	8.310	83.222
p + v + w	2.907	7.932	5.507	3.569	10.444	6.756	4.590	14.307	8.280	85.292
p + w	2.891	7.886	**5.493**	3.572	10.433	6.752	4.583	14.306	8.295	83.863
p + w + v + t	2.899	8.035	5.512	3.575	10.587	6.772	4.567	14.353	8.306	83.414
All	2.986	8.102	5.511	3.667	10.649	6.754	4.661	14.370	8.338	83.257

Table 6.33: Results with the use of different combinations of weather features in VCI

Weather Features	15 min			30 min			60 min			Avg. time
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	p/ Epoch (s)
None	**3.800**	**6.452**	**6.458**	**4.397**	**8.180**	**7.658**	**5.210**	10.399	**9.151**	120.941
p + v	3.845	6.539	6.495	4.458	8.249	7.722	5.273	**10.348**	9.179	123.201
p + v + w	3.860	6.540	6.532	4.470	8.286	7.756	5.293	10.544	9.252	118.959
p + w	3.866	6.585	6.540	4.489	8.362	7.815	5.365	10.614	9.371	123.039
p + w + v + t	3.926	6.741	6.603	4.558	8.554	7.873	5.431	10.841	9.436	119.066
All	3.992	6.802	6.635	4.658	8.745	7.906	5.431	10.872	9.546	120.363

The results observed in the three datasets are similar in the sense that adding the features does not improve the results and instead makes them slightly worse. The combinations that almost have the same performance as having no features are 'precip_rate + visibility' and 'precip_rate + wind_spd'. The lack of good results could mean that since the problem is a short-term traffic prediction, these weather dependencies don't have as much influence as in longer-term predictions.

In terms of training time required, adding these features does not add a substantial amount of time. This shows that despite the increase in the input data size, the models take roughly the same time to process it.

6.4 Global Comparisons

After making the comparisons between different approaches within the scope of the work, it makes sense to finally present how the results compare against other techniques. Tables 6.34, 6.35 and 6.36 show the results of comparing the developed approaches with baseline techniques on the three datasets: PeMS-BAY, METR-LA and VCI, respectively. In VCI the GNN-based techniques STGCN and DCRNN are not displayed since the results provided are the ones stated by the papers themselves, and because of that, there are no results for VCI. In an ideal scenario, the results with these two techniques should have been trained and tested in this work, but the code available for them was running on outdated software versions, and it was not possible to get them to work as expected originally.

Table 6.34: Comparison of the best-performing models with baselines in PeMS-BAY

Techniques	15 min			30 min			60 min		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
HA	3.401	8.169	6.693	3.401	8.169	6.693	3.401	8.169	6.693
ARIMA	1.702	3.564	3.325	2.367	5.506	4.798	3.389	8.452	6.549
VAR	1.786	3.874	3.179	2.356	5.125	4.323	2.935	6.764	5.617
DCRNN	1.380	**2.900**	**2.950**	**1.740**	**3.900**	**3.970**	**2.070**	**4.900**	**4.740**
STGCN	**1.360**	**2.900**	2.960	1.810	4.170	4.270	2.490	5.790	5.690
ChebGRUAtt	1.688	3.705	3.230	2.233	5.207	4.375	2.909	7.109	5.533

Table 6.35: Comparison of the best-performing models with baselines in METR-LA

Techniques	15 min			30 min			60 min		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
HA	11.013	23.340	14.737	11.013	23.340	14.737	11.013	23.340	14.737
ARIMA	3.990	9.600	8.210	5.150	12.700	10.450	6.900	17.400	13.230
VAR	4.420	10.200	7.890	5.410	12.700	10.450	6.520	10.110	15.800
DCRNN	**2.770**	**7.300**	**5.380**	**3.150**	8.800	6.450	**3.600**	**10.500**	**7.590**
STGCN	2.870	7.400	5.540	3.480	**9.400**	6.840	4.450	11.800	8.410
ChebGRUAtt	2.925	7.486	5.503	3.547	9.538	**6.318**	4.446	12.481	8.006
ChebGRUAtt + Missing Data	2.877	7.832	5.500	3.535	10.349	6.748	4.523	14.159	8.266

In PeMS-BAY the results of the developed models fall short of the expectations compared to the other approaches. ChebGRUAtt has worse results than the other GNN-based approaches in

Table 6.36: Comparison of the best-performing models with baselines in VCI

Techniques	15 min			30 min			60 min		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
HA	9.234	17.652	12.109	9.234	17.652	12.109	9.234	17.652	12.109
ARIMA	4.283	9.781	8.346	5.097	12.540	10.002	6.764	15.458	13.166
VAR	4.443	10.010	8.590	5.410	12.626	10.237	6.638	15.295	13.450
ChebGRUAtt	**3.785**	**6.432**	6.594	4.451	8.378	7.928	5.382	10.963	9.595
ChebGRUAtt + Missing Data	3.825	6.890	**6.430**	**4.417**	**8.174**	**7.656**	**5.221**	**10.367**	**9.119**

almost all cases and even when compared to VAR in some instances. Since the results of PeMS-BAY are much better overall than in the other datasets, the room for improvement is smaller, and time-series techniques have results more on par with the GNN-based techniques. This can be from the lack of missing data on this dataset or the overall quality of the data present in it. In datasets like this, where time-series data performs well, it actually can trigger the question of whether deep-learning-based techniques (such as GNNs) are worth employing, given that the computation power and time required for training these models is substantial, especially when compared with the time-series algorithms employed here.

In METR-LA however, it outperforms all time-series-based approaches in all metrics and horizons, and it also outperforms STGCN in some cases. DCRNN, however still has the best results overall. In terms of VCI, it is only possible to compare with the time-series-based approaches, and it is significantly better than those for all metrics and horizons.

6.5 Results Discussion

In sum, the empirical experiments explored and answered several different questions:

- What combinations of spatial and temporal mechanisms work best for this problem?

  It was found that the combination of Chebyshev Spectral Graph Convolution and Gated Recurrent Unit was the best model across all three datasets. It was possible to see the way the explored models perform relative to each other was the same across all datasets, i.e. if model X is better than model Y in dataset Z, it also is better in dataset W.

- What hyperparameter settings are better?

  It was found that they work differently in different models, but within the same model, the results when using the same hyperparameter settings in other datasets worked more or less in the same way.

- Do missing data imputation techniques help achieve better results?

  what was discovered is that the impact the measure have depends on the amount of missing data occurring in the original dataset. The effect can be close to null if the amount is minimal. However, when the presence is more significant, the results improve with the use of imputation techniques, especially when using mean imputation and MICE.

- Can the models extract helpful knowledge from adding weather features and get better results?

  Their impact on the results was negative, pointing against the use of those features.

- How do the models compare to existing techniques?

  Compared to other techniques, the results achieved are not groundbreaking, but they point in the direction that GNNs are a good approach to this traffic prediction problem, and they can be used in further research of other mechanisms and hyperparameter configurations.

Something worth mentioning that could be seen throughout the chapter is that time constraints impacted the number of experiments carried out and the different combinations that could have been explored. There is the possibility that some of the combinations that were not explored could have led to better results than the ones presented, even if the tests chosen to do were the most theoretically promising. This is a common struggle in studying machine learning algorithms, and despite trying to minimize its impact by using multiple machines to run the experiences, it still posed an obstacle.

Chapter 7

Conclusions and Future Work

The motivation behind this research stems from the increasing demand for reliable and accurate traffic predictions in today's fast-paced world. With the increasing number of vehicles on the road, traffic congestion is becoming a widespread issue, leading to higher levels of pollution and decreased mobility. To address this problem, the field of traffic forecasting has been subject to thorough research over the years, given the complexity of the spatial and temporal dependencies of traffic data.

Approaches to traffic forecasting include various techniques that range from traditional statistical methods to Deep Learning. However, most approaches have their limitations when it comes to handling the complex temporal and spatial dependencies of graph-like traffic data. This is where Graph Neural Networks come into the picture. GNNs have been shown to be effective in handling complex relationships between data and have been successfully applied to various problems.

The objective of this dissertation consisted of applying GNNs to the problem of traffic forecasting and evaluating their performance against other frequently used methods, considering missing data handling and the incorporation of external factors as additional methods to try to improve results.

An important contribution of this work was the standardization of the process of generating the input data for the GNN models, in order to easily allow for the training and testing in different datasets.

The empirical evaluation showed that the explored GNNs outperform several commonly used time-series-based models in the literature, affirming the leverage that these models can have. However, they did not outperform other GNN-based techniques existing in the literature, meaning that some more tuning to the architectures and parameters should be done.

Another contribution of this dissertation is the study of the impact of using missing data imputation techniques on prediction performance. It was shown that GNNs can benefit from these techniques, especially when used with VCI and METR-LA datasets. The research also demonstrated that the incorporation of weather conditions does not produce a positive impact on the results, contrary to the initial hypothesis.

Moving forward, the potential applications of GNNs in Intelligent Transportation Systems are vast, and further research is needed to explore their full potential. This work opens several directions for future work, such as a more intensive study of hyperparameters and layer architectures to try to achieve better results.

On top of that, a broader study of sources of external data, such as social events happening close to the road network, social media mentions of congestion or even accidents happening on the network could be used to enrich the models.

References

Alam, I., Ahmed, M. F., Alam, M., Ulisses, J., Farid, D. M., Shatabda, S., & Rossetti, R. J. F. (2017). Pattern mining from historical traffic big data. In *2017 ieee region 10 symposium (tensymp)* (p. 1-5). doi: 10.1109/TENCONSpring.2017.8070031

Alam, I., Farid, D. M., & Rossetti, R. J. F. (2019). The prediction of traffic flow with regression analysis. In A. Abraham, P. Dutta, J. K. Mandal, A. Bhattacharya, & S. Dutta (Eds.), *Emerging technologies in data mining and information security* (pp. 661–671). Singapore: Springer Singapore.

Alsolami, B., Mehmood, R., & Albeshri, A. (2020). Hybrid Statistical and Machine Learning Methods for Road Traffic Prediction: A Review and Tutorial. In R. Mehmood, S. See, I. Katib, & I. Chlamtac (Eds.), *Smart Infrastructure and Applications: Foundations for Smarter Cities and Societies* (pp. 115–133). Springer International Publishing. doi: 10.1007/978-3-030-13705-2_5

Ang, K. L.-M., Seng, J. K. P., Ngharamike, E., & Ijemaru, G. K. (2022). Emerging Technologies for Smart Cities’ Transportation: Geo-Information, Data Analytics and Machine Learning Approaches. *ISPRS International Journal of Geo-Information*, *11*(2). doi: 10.3390/ijgi11020085

Angarita-Zapata, J. S., Masegosa, A. D., & Triguero, I. (2019). A taxonomy of traffic forecasting regression problems from a supervised learning perspective. *IEEE Access*, *7*, 68185-68205. doi: 10.1109/ACCESS.2019.2917228

Baldassarre, F., & Azizpour, H. (2019). Explainability techniques for graph convolutional networks. *ArXiv*, *abs/1905.13686*.

Barros, J., Araujo, M., & Rossetti, R. J. F. (2015, June). Short-term real-time traffic prediction methods: A survey. In *2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)* (pp. 132–139). doi: 10.1109/MTITS.2015.7223248

Bhanja, S., & Das, A. (2019, January). *Impact of Data Normalization on Deep Neural Network for Time Series Forecasting.* arXiv. doi: 10.48550/arXiv.1812.05519

Bokaba, T., Doorsamy, W., & Paul, B. S. (2022, January). A Comparative Study of Ensemble Models for Predicting Road Traffic Congestion. *Applied Sciences*, *12*(3), 1337. doi: 10.3390/app12031337

Bui, K.-H. N., Cho, J., & Yi, H. (2022, February). Spatial-temporal graph neural network for traffic

forecasting: An overview and open research issues. *Applied Intelligence*, *52*(3), 2763–2774. doi: 10.1007/s10489-021-02587-w

Cai, L., Zhang, Z., Yang, J., Yu, Y., Zhou, T., & Qin, J. (2019, December). A noise-immune Kalman filter for short-term traffic flow forecasting. *Physica A: Statistical Mechanics and its Applications*, *536*, 122601. doi: 10.1016/j.physa.2019.122601

Cao, S., Wu, L., Zhang, R., Li, J., & Wu, D. (2022, July). Capturing Local and Global Spatial-Temporal Correlations of Spatial-Temporal Graph Data for Traffic Flow Prediction. In *2022 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8). doi: 10.1109/IJCNN55064.2022.9892616

Cao, W., Wang, D., Li, J., Zhou, H., Li, L., & Li, Y. (2018). BRITS: Bidirectional Recurrent Imputation for Time Series. In *Advances in Neural Information Processing Systems* (Vol. 31). Curran Associates, Inc. Retrieved 2023-01-31, from https://proceedings.neurips.cc/paper/2018/hash/734e6bfcd358e25ac1db0a4241b95651-Abstract.html

Castro, P. S., Zhang, D., & Li, S. (2012). Urban Traffic Modelling and Prediction Using Large Scale Taxi GPS Traces. In J. Kay, P. Lukowicz, H. Tokuda, P. Olivier, & A. Krüger (Eds.), *Pervasive Computing* (pp. 57–72). Berlin, Heidelberg: Springer. doi: 10.1007/978-3-642-31205-2_4

Che, Z., Purushotham, S., Cho, K., Sontag, D. A., & Liu, Y. (2016). Recurrent neural networks for multivariate time series with missing values. *CoRR*, *abs/1606.01865*. Retrieved from http://arxiv.org/abs/1606.01865

Chen, C., Li, K., Teo, S. G., Zou, X., Wang, K., Wang, J., & Zeng, Z. (2019, July). Gated Residual Recurrent Graph Neural Networks for Traffic Prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 485–492. doi: 10.1609/aaai.v33i01.3301485

Chen, D., Yan, X., Liu, X., Li, S., Wang, L., & Tian, X. (2021). A multiscale-grid-based stacked bidirectional gru neural network model for predicting traffic speeds of urban expressways. *IEEE Access*, *9*, 1321-1337. doi: 10.1109/ACCESS.2020.3034551

Chen, X., Cai, X., Liang, J., & Liu, Q. (2018). Ensemble learning multiple lssvr with improved harmony search algorithm for short-term traffic flow forecasting. *IEEE Access*, *6*, 9347-9357. doi: 10.1109/ACCESS.2018.2805299

Chen, X., Chen, Y., & He, Z. (2018, March). *Urban traffic speed dataset of guangzhou, china.* Zenodo. doi: 10.5281/zenodo.1205229

Cui, Z. (2023, January). *Seattle Inductive Loop Detector Dataset V.1 (2015).* Retrieved 2023-01-30, from https://github.com/zhiyongc/Seattle-Loop-Data

Cui, Z., Henrickson, K. C., Ke, R., & Wang, Y. (2018). Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE Transactions on Intelligent Transportation Systems*, *21*, 4883-4894. doi: 10.48550/arXiv.1802.07007

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *CoRR*. doi: 10.48550/arXiv.1606.09375

Ge, L., Li, H., Liu, J., & Zhou, A. (2019, June). Temporal Graph Convolutional Networks for Traffic Speed Prediction Considering External Factors. In *2019 20th IEEE International Conference on Mobile Data Management (MDM)* (pp. 234–242). doi: 10.1109/MDM.2019.00-52

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017, July). Convolutional Sequence to Sequence Learning. *CoRR*, 1243–1252. Retrieved 2023-06-19, from https://proceedings.mlr.press/v70/gehring17a.html

Guo, S., Lin, Y., Feng, N., Song, C., & Wan, H. (2019, July). Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 922–929. Retrieved from https://ojs.aaai.org/index.php/AAAI/article/view/3881 doi: 10.1609/aaai.v33i01.3301922

Hamilton, W. L. (2020). *Graph Representation Learning*. Cham: Springer International Publishing. doi: 10.1007/978-3-031-01588-5

He, Z., Chow, C.-Y., & Zhang, J.-D. (2019, June). STCNN: A Spatio-Temporal Convolutional Neural Network for Long-Term Traffic Prediction. In *2019 20th IEEE International Conference on Mobile Data Management (MDM)* (pp. 226–233). doi: 10.1109/MDM.2019.00-53

Hu, J., Lin, X., & Wang, C. (2022, July). DSTGCN: Dynamic Spatial-Temporal Graph Convolutional Network for Traffic Prediction. *IEEE Sensors Journal*, *22*(13), 13116–13124. doi: 10.1109/JSEN.2022.3176016

Irawan, K., Yusuf, R., & Prihatmanto, A. S. (2020, December). A Survey on Traffic Flow Prediction Methods. In *2020 6th International Conference on Interactive Digital Media (ICIDM)* (pp. 1–4). doi: 10.1109/ICIDM51048.2020.9339675

Ismiguzel, I. (2022, May). *Imputing Missing Data with Simple and Advanced Techniques.* Retrieved 2023-06-10, from https://towardsdatascience.com/imputing-missing-data-with-simple-and-advanced-techniques-f5c7b157fb87

Jensen, M. (2019, August). *Data Leakage and how to avoid it.* Retrieved 2023-06-12, from https://neurospace.io/blog/2019/08/data-leakage-and-how-to-avoid-it/

Jiang, R., Yin, D., Wang, Z., Wang, Y., Deng, J., Liu, H., ... Shibasaki, R. (2021). Dl-traff: Survey and benchmark of deep learning models for urban traffic prediction. In *Proceedings of the 30th acm international conference on information & knowledge management* (p. 4515–4525). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3459637.3482000

Jiang, W., & Luo, J. (2021). Big data for traffic estimation and prediction: A survey of data and tools. *CoRR*, *abs/2103.11824*. doi: 10.48550/arXiv.2103.11824

Jiang, W., & Luo, J. (2022, November). Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, *207*, 117921. doi: 10.1016/j.eswa.2022.117921

Karunasingha, D. S. K. (2022, March). Root mean square error or mean absolute error? Use their ratio as well. *Information Sciences*, *585*, 609–629. doi: 10.1016/j.ins.2021.11.036

Kipf, T. N., & Welling, M. (2017, February). *Semi-Supervised Classification with Graph Convolutional Networks.* arXiv. doi: 10.48550/arXiv.1609.02907

Kołodziej, J., Hopmann, C., Coppa, G., Grzonka, D., & Widłak, A. (2022). Intelligent Transportation Systems – Models, Challenges, Security Aspects. In J. Kołodziej, M. Repetto, & A. Duzha (Eds.), *Cybersecurity of Digital Service Chains: Challenges, Methodologies, and Tools* (pp. 56–82). Cham: Springer International Publishing. doi: 10.1007/978-3-031-04036-8_3

Lau, S. (2017, August). *Learning Rate Schedules and Adaptive Learning Rate Methods for Deep Learning.* Retrieved 2023-03-28, from https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1

Laña, I., Del Ser, J., Velez, M., & Vlahogianni, E. (2018, July). Road Traffic Forecasting: Recent Advances and New Challenges. *IEEE Intelligent Transportation Systems Magazine*, *10*, 93–109. doi: 10.1109/MITS.2018.2806634

Lee, K., Eo, M., Jung, E., Yoon, Y., & Rhee, W. (2021). Short-term traffic prediction with deep neural networks: A survey. *IEEE Access*, *9*, 54739-54756. doi: 10.1109/ACCESS.2021.3071174

Lee, K., & Rhee, W. (2022). DDP-GCN: Multi-graph convolutional network for spatiotemporal traffic forecasting. *Transportation Research Part C: Emerging Technologies*, *134*, 103466. doi: https://doi.org/10.1016/j.trc.2021.103466

Lee, S., & b. Fambro, D. (1999). Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting. *Transportation Research Record*, *1678*, 179 - 188.

Li, Y., Mensi, F., & Yu, R. (2023, June). *Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting.* Retrieved 2023-06-06, from https://github.com/liyaguang/DCRNN

Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2018). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International conference on learning representations.* doi: 10.48550/arXiv.1707.01926

Liu, J., Wu, N., Qiao, Y., & Li, Z. (2021, 01). A scientometric review of research on traffic forecasting in transportation. *IET Intelligent Transport Systems*, *15*. doi: 10.1049/itr2.12024

Liu, Z., & Zhou, J. (2020). *Introduction to Graph Neural Networks*. Cham: Springer International Publishing. doi: 10.1007/978-3-031-01587-8

Lopes, J., Bento, J., Huang, E., Antoniou, C., & Ben-Akiva, M. (2010, September). Traffic and mobility data collection for real-time applications. In *13th International IEEE Conference on Intelligent Transportation Systems* (pp. 216–223). doi: 10.1109/ITSC.2010.5625282

Lu, H., Huang, D., Song, Y., Jiang, D., Zhou, T., & Qin, J. (2020, Septembet). St-trafficnet: A spatial-temporal deep learning network for traffic forecasting. *Electronics*. doi: 10.3390/electronics9091474

Lu, J., Li, B., Li, H., & Al-Barakani, A. (2021). Expansion of city scale, traffic modes, traffic

congestion, and air pollution. *Cities*, *108*, 102974. doi: 10.1016/j.cities.2020.102974

Ma, T., Antoniou, C., & Toledo, T. (2020). Hybrid machine learning algorithm and statistical time series model for network-wide traffic forecast. *Transportation Research Part C-emerging Technologies*, *111*, 352-372.

Ma, X., Dai, Z., He, Z., Ma, J., Wang, Y., & Wang, Y. (2017). Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction. *Sensors (Basel, Switzerland)*, *17*.

Mori, U., Mendiburu, A., Álvarez, M., & Lozano, J. A. (2015). A review of travel time estimation and forecasting for Advanced Traveller Information Systems. *Transportmetrica A Transport Science*, *11*(2), 119–157. doi: https://doi.org/10.1080/23249935.2014.932469

O'Connor, R. (2021, December). *PyTorch vs TensorFlow in 2023.* Retrieved 2023-03-27, from https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2023/

Pavlyuk, D. (2017, January). Short-term Traffic Forecasting Using Multivariate Autoregressive Models. *Procedia Engineering*, *178*, 57–66. doi: 10.1016/j.proeng.2017.01.062

PredictHQ. (2023a). *Events api.* https://predicthq.com. Retrieved 2023-02-16, from https://predicthq.com

PredictHQ. (2023b). *Events - Technical Documentation - PredictHQ.* Retrieved 2023-02-16, from https://docs.predicthq.com/resources/events#event-fields

Raschka, S., Patterson, J., & Nolet, C. (2020, April). Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence. *Information*, *11*(4), 193. Retrieved 2023-03-27, from https://www.mdpi.com/2078-2489/11/4/193 doi: 10.3390/info11040193

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press. doi: 10.1017/CBO9780511812651

Rozemberczki, B. (2023). *PyTorch Geometric Temporal Documentation.* Retrieved 2023-03-27, from https://pytorch-geometric-temporal.readthedocs.io/en/latest/index.html

Sanchez-Lengeling, B., Reif, E., Pearce, A., & Wiltschko, A. B. (2021, September). A Gentle Introduction to Graph Neural Networks. *Distill*, *6*(9), e33. Retrieved 2023-02-04, from https://distill.pub/2021/gnn-intro doi: 10.23915/distill.00033

Schörner, P., Hubschneider, C., Härtl, J., Polley, R., & Zöllner, J. M. (2019, October). Grid-Based Micro Traffic Prediction using Fully Convolutional Networks. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)* (pp. 4540–4547). doi: 10.1109/ITSC.2019.8917263

Sergios Karagiannakos, A. S. (2021, Sep). *Best Graph Neural Network architectures: GCN, GAT, MPNN and more.* Retrieved 2022-11-03, from https://theaisummer.com/gnn-architectures/

Shahid, N., Shah, M. A., Khan, A., Maple, C., & Jeon, G. (2021, September). Towards greener smart cities and road traffic forecasting using air pollution data. *Sustainable Cities and Society*, *72*, 103062. doi: 10.1016/j.scs.2021.103062

Shi, R., & Du, L. (2022, October). Multi-Section Traffic Flow Prediction Based on MLR-LSTM Neural Network. *Sensors (Basel, Switzerland)*, *22*(19), 7517. doi: 10.3390/s22197517

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., & Vandergheynst, P. (2013, May). The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains. *IEEE Signal Processing Magazine*, *30*(3), 83–98. doi: 10.1109/MSP.2012.2235192

Shuvo, M. A. R., Zubair, M., Purnota, A. T., Hossain, S., & Hossain, M. I. (2021, January). Traffic Forecasting using Time-Series Analysis. In *2021 6th International Conference on Inventive Computation Technologies (ICICT)* (pp. 269–274). doi: 10.1109/ICICT50816.2021.9358682

Smith, B. L., Williams, B. M., & Keith Oswald, R. (2002, August). Comparison of parametric and nonparametric models for traffic flow forecasting. *Transportation Research Part C: Emerging Technologies*, *10*(4), 303–321. doi: 10.1016/S0968-090X(02)00009-8

State of California. (2023). *PeMS Data Source | Caltrans*. https://dot.ca.gov/programs/traffic-operations/mpr/pems-source.

Sun, H., Liu, H. X., Xiao, H., He, R. R., & Ran, B. (2003). Use of local linear regression model for short-term traffic forecasting. *Transportation Research Record*, *1836*(1), 143-150. doi: 10.3141/1836-18

Tedjopurnomo, D. A., Bao, Z., Zheng, B., Choudhury, F. M., & Qin, A. K. (2022). A survey on modern deep neural network for traffic prediction: Trends, methods and challenges. *IEEE Transactions on Knowledge and Data Engineering*, *34*(4), 1544-1561. doi: 10.1109/TKDE.2020.3001195

United Nations, U. (2017, September). *Rapid urbanisation: opportunities and challenges to improve the well-being of societies.* https://hdr.undp.org/content/rapid-urbanisation-opportunities-and-challenges-improve-well-being-societies.

V, R., & S, G. V. (2022, July). Hybrid Time-Series Forecasting Models for Traffic Flow Prediction. *Promet*, *34*(4), 537–549. Retrieved 2023-01-15, from https://traffic2.fpz.hr/index.php/PROMTT/article/view/15 doi: 10.7307/ptt.v34i4.3998

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *ArXiv*, *abs/1609.03499*.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio', P., & Bengio, Y. (2017). *Graph attention networks* (Vol. abs/1710.10903).

Vlahogianni, E., Golias, J., & Karlaftis, M. (2004, 09). Short-term traffic forecasting: Overview of objectives and methods. *Transport Reviews*, *24*, 533-557. doi: 10.1080/0144164042000195072

Vlahogianni, E. I., Karlaftis, M. G., & Golias, J. C. (2014, June). Short-term traffic forecasting: Where we are and where we're going. *Transportation Research Part C: Emerging Technologies*, *43*, 3–19. doi: 10.1016/j.trc.2014.01.005

Wang, Y., Li, L., & Xu, X. (2017, 10). A piecewise hybrid of arima and svms for short-term traffic

flow prediction. In *Neural information processing* (p. 493-502). doi: 10.1007/978-3-319-70139-4_50

Weatherbit. (2023a). *Weatherbit - Free Weather API.* Retrieved 2023-02-21, from https://www.weatherbit.io/

Weatherbit. (2023b). *Weatherbit | Sub-Hourly Weather History API Documentation.* https://www.weatherbit.io/api/weather-history-subhourly. Retrieved 2023-02-17, from https://www.weatherbit.io/api/weather-history-subhourly

WHO, W. H. O. (2022). *Billions of people still breathe unhealthy air: new WHO data.* Retrieved 2022-11-03, from https://www.who.int/news/item/04-04-2022-billions-of-people-still-breathe-unhealthy-air-new-who-data

Wu, Y., & Tan, H. (2016, December). *Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework.* arXiv. doi: 10.48550/arXiv.1612.01022

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021a). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(1), 4-24. doi: 10.1109/TNNLS.2020.2978386

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021b, January). A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(1), 4–24. doi: 10.1109/TNNLS.2020.2978386

Wu, Z., Pan, S., Long, G., Jiang, J., & Zhang, C. (2019). Graph wavenet for deep spatial-temporal graph modeling. In *International joint conference on artificial intelligence.*

Xie, Q., Guo, T., Chen, Y., Xiao, Y., Wang, X., & Zhao, B. Y. (2019). "how do urban incidents affect traffic speed?" a deep graph convolutional network for incident-driven traffic speed prediction. *ArXiv*, *abs/1912.01242*.

Yan, H., Fu, L., Qi, Y., Yu, D.-J., & Ye, Q. (2022, August). Robust ensemble method for short-term traffic flow prediction. *Future Generation Computer Systems*, *133*, 395–410. doi: 10.1016/j.future.2022.03.034

Yao, H., Tang, X., Wei, H., Zheng, G., Yu, Y., & Li, Z. (2018, March). Modeling Spatial-Temporal Dynamics for Traffic Prediction. *ArXiv*. Retrieved 2023-01-18, from https://www.semanticscholar.org/paper/Modeling-Spatial-Temporal-Dynamics-for-Traffic-Yao-Tang/dc5d1b5a1165aab6d042f48ea7b068a90a900d68

Ye, J., Xue, S., & Jiang, A. (2022, June). Attention-based spatio-temporal graph convolutional network considering external factors for multi-step traffic flow prediction. *Digital Communications and Networks*, *8*(3), 343–350. doi: 10.1016/j.dcan.2021.09.007

Ye, J., Zhao, J., Ye, K., & Xu, C. (2022, May). How to build a graph-based deep learning architecture in traffic domain: A survey. *IEEE Transactions on Intelligent Transportation Systems*, *23*(5), 3904-3924. doi: 10.1109/TITS.2020.3043250

Yin, X., Wu, G., Wei, J., Shen, Y., Qi, H., & Yin, B. (2022, June). Deep Learning on Traffic Prediction: Methods, Analysis, and Future Directions. *IEEE Transactions on Intelligent Transportation Systems*, *23*(6), 4927–4943. doi: 10.1109/TITS.2021.3054840

Yu, B., Yin, H., & Zhu, Z. (2018, July). Spatio-Temporal Graph Convolutional Networks: A

Deep Learning Framework for Traffic Forecasting. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence* (pp. 3634–3640). doi: 10.24963/ijcai.2018/505

Yu, T., Yin, H., & Zhu, Z. (2019). St-unet: A spatio-temporal u-network for graph-structured time series modeling. *ArXiv*, *abs/1903.05631*.

Yuan, H., & Li, G. (2021, March). A Survey of Traffic Prediction: from Spatio-Temporal Data to Intelligent Transportation. *Data Science and Engineering*, *6*(1), 63–85. doi: 10.1007/s41019-020-00151-z

Zhang, Q., Chang, J., Meng, G., Xiang, S., & Pan, C. (2020, April). Spatio-Temporal Graph Structure Learning for Traffic Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(01), 1177–1185. Retrieved 2022-10-26, from https://ojs.aaai.org/index.php/AAAI/article/view/5470 doi: 10.1609/aaai.v34i01.5470

Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., ... Li, H. (2020, September). T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction. *IEEE Transactions on Intelligent Transportation Systems*, *21*(9), 3848–3858. doi: 10.1109/TITS.2019.2935152

Zhao, Y., Zhang, H., An, L., & Liu, Q. (2018, December). Improving the approaches of traffic demand forecasting in the big data era. *Cities*, *82*, 19–26. doi: 10.1016/j.cities.2018.04.015

Zheng, C., Fan, X., Wang, C., & Qi, J. (2019). Gman: A graph multi-attention network for traffic prediction. *ArXiv*, *abs/1911.08415*.

Zhong, W., Suo, Q., Jia, X., Zhang, A., & Su, L. (2021, July). Heterogeneous Spatio-Temporal Graph Convolution Network for Traffic Forecasting with Missing Values. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)* (pp. 707–717). DC, USA: IEEE. doi: 10.1109/ICDCS51616.2021.00073

Zhou, K., Liu, Z., Duan, K., & Hu, X. (2022). Graph Neural Networks: AutoML. In L. Wu, P. Cui, J. Pei, & L. Zhao (Eds.), *Graph Neural Networks: Foundations, Frontiers, and Applications* (pp. 371–389). Singapore: Springer Nature. doi: 10.1007/978-981-16-6054-2_17

Zhou, X., Shen, Y., & Huang, L. (2019). *Revisiting flow information for traffic prediction.*

Zhu, J., Song, Y., Zhao, L., & Li, H. (2020, June). *A3T-GCN: Attention Temporal Graph Convolutional Network for Traffic Forecasting.* arXiv. doi: 10.48550/arXiv.2006.11583