UNIVERSITY OF MILANO-BICOCCA
DEPARTMENT OF COMPUTER SCIENCE, SYSTEMS AND COMMUNICATION
PH.D. PROGRAM OF COMPUTER SCIENCE

Listening to the City: Artificial Intelligence meets Smart Mobility

Supervisor: Prof. Michele Ciavotta
Ph.D. Tutor: Prof. Alberto Dennunzio
Ph.D. Coordinator: Prof. Leonardo Mariani

Final Report's Summary of:
STEFANO FIORINI
STUDENT NUMBER 778379

Academic Year 2021 - 2022

Abstract

Cities are constantly evolving due to the changes and interactions with their citizens. They are complex systems made up of individuals, vehicles, and infrastructures that form the heart of modern life and social development. A key factor driving this growth is the advancement of mobility towards a more intelligent and technological scenario, also known as Smart Mobility. This concept involves moving from a traditional transportation system to a more advanced one, where an intelligent infrastructure fabric connects stakeholders and entities to provide efficient, smart, and sustainable solutions. Researchers in this field place a strong emphasis on understanding and identifying individuals' displacement patterns, in order to enhance the services and, more in general, the livability of cities. The advent of Data Science has revealed the importance of developing and applying Artificial Intelligence techniques in the creation of automated and intelligent city systems. Although AI is increasingly used in Smart Mobility, there are still many challenges to be addressed. For this reason, in this thesis, AI and Transportation Science are combined to develop solutions to mobility-related problems by extracting knowledge from mobility data and developing sophisticated algorithms. We show, through extensive experimentation, that the proposed solutions effectively improve the understanding and application of Artificial Intelligence in mobility problems. In particular, we initially focus on data processing and knowledge extraction. In order to explain the adoption of shared mobility vehicles, we propose the use of geographic (distance from center, walkability, concentration of places) and demographic features (education index). Additionally, we present a multi-objective optimization algorithm to identify behavioral communities, taking into account shared mobility usage patterns, distance between areas, and structural information (via Map Embeddings). We also explore the theoretical and architectural definition of Deep Learning techniques for solving real-world mobility problems. We propose two Deep Learning architectures, 3D-CLoST and STREED-Net, which use innovative solutions to better extract spatio-temporal patterns from mobility data, and enable us to improve the state of the art in the short-term flow prediction problem. Furthermore, to extend the applicability of spectral-based Graph Convolutional Networks (GCNs) to mobility problems defined by means of directed weighted graphs, we introduce SigMaNet, which can handle both undirected and directed graphs with weights of any sign or magnitude. The cornerstone of SigMaNet is the new *Sign-Magnetic Laplacian* matrix. Finally, we address on-street parking prediction by assessing the performance of various methods, including statistical models, GCNs, and Convolutional Neural Networks (CNNs), on two indicators, the average parking time and the average number of vehicles parked simultaneously.

Contents

List of Figures

List of Tables

List of acronyms

ICT Information and Communication Technologies

SC Smart City

AI Aritficial Intelligence

ANN Artifical Neural Network

CNN Convolutional Neural Network

GCN Graph Convolutional Network

DNN Deep Neural Network

PCA Principal Component Analysis

POI Point of Interest

BN Batch Normalization

SM Smart Mobility

STREED-Net Spatio Temporal REsidual Encoder-Decoder Network

3D-CLoST 3D Convolution LSTM on Spatio - Temporal

GRU Gated Recurrent Unit

RMSE Root Mean Square Error

MAPE Mean Absolute Percentage Error

APE Absolute Percentage Error

DSBM Direct Stochastic Block Model

IPA Intelligent Parking Assistant

WSN Wireless Sensor Network

OBU On Board Unit

1 Introduction

1.1 Smart City

Nowadays, almost 55% of the world's population lives in urban areas, and the number is expected to increase to 66% by 2030 [189]. As the city grows, new problems arise (e.g., traffic congestion, waste management, pollution, parking allocation) and resources are limited. Therefore, working on adapting the city to current (and future) needs is a priority for all of us, and researchers are no exception. The urgency to make the city a more suitable place for quality living is giving rise to many initiatives around the world [186], from city councils to companies to research laboratories. People from different disciplines, cultures, backgrounds and interests are finding common ground: creating Smart City (SC).

The smart city transition has a significant positive impact on three different areas: Economic, Environmental, and Social. In particular:

- **Economic Impact.** *i)* Overall economic growth of 5% per year, nearly 20 trillion Dollar in a decade [95],and *ii)* cooperation between the public and private systems to make the services made available to citizens more efficient [49]. For example, the use of carsharing results in average savings of $154 to $435 per month for a household, compared to the cost of using a private vehicle [163].
- **Environmental Impact.** *i)* 10-15% reduction in GHG emissions per year [95], and *ii)* 27-43% decrease in annual VMT (Vehicle Miles Traveled) of households [163].
- **Social Impact.** Reduction in travel time by up to 40% resulting in saving billions of hours lost due to commuting and congestion [25].

The concept of SC emerged in recent years as a way to improve the functioning of cities through the use of information and communication technologies. The idea of a smart city has been described using various metaphors, but it is often seen as a large organic system [133]. This term is becoming more widespread (more cities are being labeled as smart every day), but it is still considered a work-in-progress concept and there is no

universally accepted definition [40, 126]. However, a common one describes it as the "Effective integration of physical, digital and human systems in the built environment to deliver a sustainable, prosperous and inclusive future for its citizen" [139]. Moreover, multiple expressions have been created to refer to the same concept, but with slightly different perspectives, e.g. digital city [87] or intelligent city [101]. Even worse, the concept of Smart Cities has limited research and understanding of its implications, making it difficult to assess if advancements are being made or if important opportunities for improvement are being missed [64].

When discussing the Smart City, it is important to take into consideration a multitude of components. However, there are certain fundamental characteristics that are consistently present. These include:

- **Technology**. These can be sensors, networks, and other types of technology used to collect data and monitor city operations.
- **Infrastructure.** It may include items such as transportation systems, energy systems and other types of infrastructure used to support city operations.
- **Governance.** It may include policies, regulations, and other types of governance mechanisms used to manage city operations.
- **Sustainability.** It can include items such as renewable energy, green buildings and other sustainable practices used to make the city more environmentally friendly.
- **Quality of Life.** It can include aspects such as public safety, education and other factors that contribute to the overall quality of life of city residents.

The concept of Smart City incorporates the use of Information and Communication Technologies (ICT) and network-connected physical devices to enable city officials to interact directly with infrastructure, monitoring what is happening in the urban environment, understanding its evolution and defining effective policies. In particular, ICTs are used to analyze urban services in order to optimize mobility and safety solutions, reducing their costs and consumption of resources, targeting sustainability and ecological measures directed toward energy conservation. Data in smart cities are collected from various sources, including sensors placed in buildings, inside transport vehicles, in the streets, or from technological devices used by citizens, such as smartphones and tablets.

1.1.1 Implementing Data-Driven Smart Cities

Modern cities are experiencing significant technological and operational changes, and Data Science is driving the changes in the Fourth Industrial Revolution (Industry 4.0) [155]. The combination of these two entities gives rise to the Data-Driven Smart City [156], also known as Urban Computing. It refers to the complex process of gathering, processing, and analyzing large amounts of heterogeneous data from multiple sources, in order to address the challenges that cities present, such as air pollution, consumption of energy sources, and traffic congestion [228].

Urban computing is an interdisciplinary field that intersects several disciplines, including computer science, civil engineering, economics, and sociology. Its goal, as shown in Figure 1.1, is to create innovative solutions with significant positive impacts on three different factors: environment, citizens' quality of life, and city management.

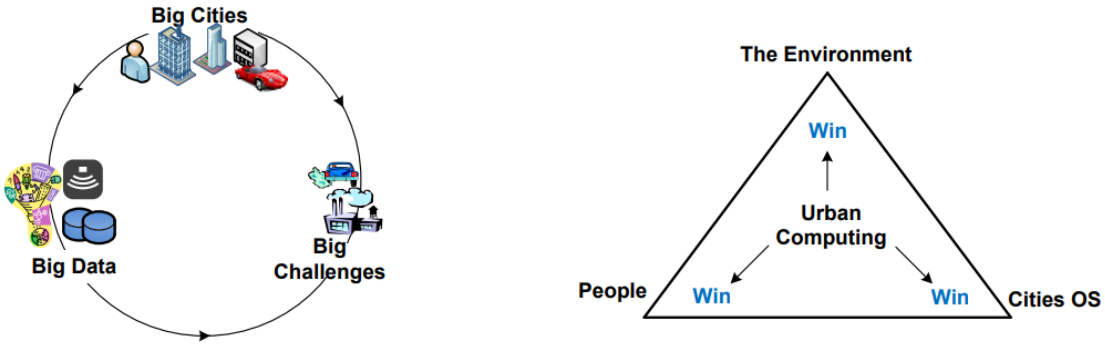


Figure 1.1 Life cycle and motivation of the urban computing [228]

Challenges

As previously stated, urban computing is a multifaceted subject that encompasses various facets. The principal challenges encountered in each one are outlined below [228]:

- **Urban sensing and data acquisition**. Firstly, one must be able to collect the necessary data in a continuous, non-intrusive and city-scale manner. Meeting all these requirements turns out to be a daunting task, as many problems can arise during the data acquisition phase. For example, if the data comes from personal mobile devices, such as smartphones, the aspects of saving or managing battery consumption and regulations related to privacy are not to be underestimated. An additional challenge related to this type of data is the lack of control over the frequency at which individuals share information. This makes data management more complex compared to working with data generated by sensors, where the sampling interval can be set. Additionally, personal data often appear unevenly distributed on the urban scale, with some areas being highly active and others being relatively inactive. Finally, these data are unstructured, meaning they are not conformed to a standard format, and are typically noisy.
- **Management of heterogeneous data**. Managing heterogeneous data is essential for finding solutions to city challenges. For example, to monitor air pollution, one might want to simultaneously take into account traffic flow, meteorology, and land use (presence of roads, factories, etc.). However, traditional data mining and Machine Learning algorithms usually work with only one type of data. Therefore, it is necessary to develop more elastic models suitable for the case addressed. Additionally, information learning must be both efficient and effective; in some cases, it may even be necessary to have answers in real time. Another key aspect concerns data visualization: being able to represent the key information encapsulated in the collected data is not easy, but it is essential for devising possible solutions.
- **Hybrid systems**. Integrating data from both the digital and physical worlds is crucial for the successful implementation of urban computing systems. This can be achieved by combining data from real-world sources, such as GPS, with that from a social network, creating so-called hybrid systems. However, designing and implementing these types of systems is significantly more complex compared to traditional systems.

Applications

Urban Computing has multiple applications in the city environment, including public safety, transportation, environment and urban planning. Some of the scenarios typically associated with this field are:

- **Urban computing for the environment.** Urbanization can have negative effects on the environment, specifically through increased pollution levels. Air, noise, and waste pollution are some of the main concerns. To address these issues, urban computing is used to develop sustainable and resilient urban planning strategies, with the ultimate goal of protecting the environment and improving people's lives. This includes efforts to reduce pollution and to contain the spread of pollution-related diseases.
- **Urban computing for saving energy resources.** Urbanization leads to a rise in energy consumption, creating a need for technology to track usage, improve infrastructure, and promote conservation. This includes not only monitoring the consumption of gasoline and electricity for transportation and gas for heating, but also researching ways to optimize resources.
- **Urban computing for public safety.** Data from various sources in urban contexts can greatly impact public safety. Advanced algorithms can aid in responding to environmental disasters or terrorist attacks, and even in preventing them. Research in this area includes planning measures to manage natural disasters such as earthquakes, landslides, and floods.
- **Urban computing for transportation/Smart Mobility**. The analysis of people's movements within the city plays a key role in urban computing. The study of mobility can have real-world benefits such as providing real-time optimal routes for drivers, reducing congestion, and predicting traffic using historical data. Research in this field could be highly beneficial also for taxi and vehicle-sharing services, as it could help pinpoint areas with the greatest need for transportation.

Since the focus of this thesis is on smart mobility, we are going to explore this topic more in detail below.

1.1.2 Smart Mobility

Smart Mobility (SM) is a widely studied embodiment of urban computing and is a key driver for transforming urban transportation and changing the way our cities move. It encompasses not only the physical movement of people and goods, but also the digital dissemination of information. The purpose of Smart Mobility is to connect all the resources of the city, including people, goods, and information. This objective is vital for enhancing a city's competitiveness and development, particularly as the need to move around increasingly sprawling urban areas becomes a growing concern [137]. For this reason, in recent years, there have been numerous global initiatives to ensure improvement in mobility for citizens. One notable project within this area is Big IoT [53]. This is a project under the IoT-European Platforms Initiative that aims to address interoperability issues by developing a generic, unified Web API for smart object platforms. This pilot project is implementing services and applications in several European cities:

- In Barcelona, traffic detectors are used to measure speed and count cars for a smart parking application and green route planning. Other sensors such as magnetometer roadside monitoring stations and Bluetooth/Wi-Fi antennas are also installed to monitor urban mobility and air quality.
- Wolfsburg has incorporated a citywide WLAN network to monitor public buses and bike share stations, and implemented real-time crowd management using security cameras. Human crowd detection was tested through mobile applications.
- In Berlin, real-time data on traffic volumes and speed are integrated with data on parking, bike sharing, charging stations, and public transport information.

- In Piedmont, traffic and environmental conditions are being monitored through the implementation of a dense network of sensors. Various services and applications are being developed for bicycle navigation, real-time traffic monitoring, and route planning that take into account air pollution in different areas of the city.

1.2 Data Science and Artificial Intelligence

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms, and systems to extract knowledge and insights from structured and unstructured data [48]. One of its key components is Aritficial Intelligence (AI), as it enables the development of advanced algorithms and models that can analyze and process vast amounts of data. Since AI is crucial to create an automated and intelligent city system [156] and it is a major component of this thesis, we will delve deeper into its knowledge and understanding. The study of human cognition has contributed to the emergence of several scientific disciplines, including AI. The goal of AI is to create algorithms that can mimic certain behaviors of the human brain, such as word or image recognition, logical reasoning, and problem-solving. As such, it acts as a bridge between the human mind and computer science.

The advancement of this discipline has brought significant economic benefits to humanity and has impacted all aspects of life. It has also greatly promoted social development and moved society into a new era [124]. In the early days of AI, the field focused on solving problems that were intellectually challenging for humans but relatively simple for computers, which could be described using formal mathematical rules. However, the true challenge for AI has been to solve tasks that are easy for humans to perform but difficult to formally describe, such as recognizing spoken words or faces in pictures, which we solve intuitively and automatically. There are two different types of AI: weak AI and strong AI [161]. Weak AI, also known as basic AI or level 1 AI, is designed to perform specific and limited tasks efficiently. For example, it can be programmed to recognize objects in an image or to solve computational problems. In contrast, strong AI, also known as Level 2 AI or generalized AI, refers to a form of AI that can perform a wide range of tasks and is able to adapt to new situations and environments. It is flexible and able to learn autonomously, without the need to be explicitly programmed for each task. J. Mark Bishop summarizes these two concepts by defining ”weak AI focuses on epistemic issues relating to engineering a simulation of human intelligent behavior, whereas strong AI, in seeking to engineer a computational system with all the causal power of a mind, focuses on the ontological" [22].

Artificial Intelligence has a wide range of applications in various fields such as medicine, transportation, finance, and education. It can be used to automate processes, make decisions, and provide personalized services. AI is a critical component in research and development within computer science. However, the use of AI also raises important ethical and safety concerns, such as the potential for massive unemployment due to automation and the risk of making decisions based on inaccurate or biased data. It is crucial for AI developers to consider these issues and take steps to address them responsibly.

1.2.1 Machine Learning, Artificial Neural Network and Deep Learning

Machine Learning and Artificial Neural Network, with Deep Learning at its core, are two major areas of research encompassed within the AI. Machine Learning (ML) is a method of automating the task of building analytical models to perform cognitive tasks such as object detection or natural language translation [94]. It uses algorithms that iteratively learn from problem-specific training data, allowing computers to discover hidden

insights and patterns without explicit programming. ML is particularly useful in high-dimensional data tasks such as classification, regression, and clustering. It can produce reliable and repeatable decisions by learning from previous calculations and extracting patterns from large databases. The ML field offers different classes of algorithms, each with multiple specifications and variants, including regression models, instance-based algorithms, decision trees, Bayesian methods and Artifical Neural Network (ANN). Moreover, depending on the given problem and available data, we can distinguish three types of Machine Learning training techniques (although many different subtypes have been proposed in the past few years): supervised learning, unsupervised learning and reinforcement learning.

Algorithms in the Artificial Neural Network family are highly valued due to their versatile structure, which allows them to be adapted for a wide range of applications. Inspired by the principle of information processing in biological systems, ANNs consist of mathematical representations of interconnected processing units called artificial neurons. Each connection between neurons, similar to synapses in a brain, transmits signals. whose strength can be adjusted during the training process by a weight that is continuously learned. The signals are processed by subsequent neurons only if a certain threshold, determined by an activation function, is exceeded. ANNs are typically organized into networks with multiple layers, including an input layer for receiving input data and one or more hidden layers for learning a nonlinear mapping between input and output [67]. To define the algorithm, the number of layers, neurons, and other properties such as the learning rate and activation function must be set. These elements are known as hyperparameters of the model and can be set manually or determined by an optimization routine, such as Bayesian Optimization [167].

Deep Neural Networks are composed of multiple hidden layers, organized into deeply nested network architectures, and they can use advanced operations (e.g., convolutions) or multiple activations in a neuron. These features allow deep neural networks to be fed with raw input data and automatically discover a representation needed for the corresponding learning task. As a result, Deep Neural Networks can use labeled datasets in supervised learning to train the algorithm, but can also handle unstructured data, such as text, images, and audio, and extract the relevant features for different categories of data. This reduces the need for human intervention and enables the use of larger datasets. In contrast, traditional machine learning, also known as "shallow" learning, relies more on human intervention to determine the feature set and typically requires more structured data for learning [108].

Deep Learning has become increasingly popular in recent years due to its successes in various research areas such as Computer Vision [69], Neural Language Processing [190] and Reinforcement Learning [131]. However, there are still many challenges that Deep Learning models are unable to overcome, such as the ability of relational and causal reasoning, conceptual abstraction, many other human skills, and generalization of new and unseen situations during the learning process. Additionally, these models are difficult to interpret and explain, as their decisions are based on a number of internal weights and biases that are not easily accessible or understandable to humans, making them appear as a "black box". Despite these challenges, Artificial Intelligence and Deep Learning are making significant progress in computer science and will continue to be a crucial part of research and development in the future.

1.3 Scope and Research Questions

This thesis aspires to bring together Artificial Intelligence and Transportation Science. The aim is to investigate innovative solutions to solve mobility-related problems by understanding and identifying individuals' movement patterns. This approach is expected to lead to a deeper understanding of the complex challenges in mobility and to the development of more effective and efficient solutions. In the thesis, we first examine the application of

regression and clustering techniques in the world of mobility, with a focus on data processing and knowledge creation. We then explore the theoretical and architectural definition of new Deep Learning solutions applied to solving real problems related to Smart Mobility, such as predicting travel flows. In particular, great attention is given to how extracting relevant information from data can improve the performance of predictions.
The research in this thesis has been guided by several research questions:

Research Question 1. *Can we predict the level of adoption of shared vehicles in a city, by analyzing and identifying the factors that drive their usage?*

To answer this question, in Chapter 2 and in Fiorini et al. [61], we devise and examine three features concerning the geographic characteristics (distance from center, walkability, concentration of places) and one about the population (education index). Our findings demonstrate that these features can serve as indicators/drivers for predicting the adoption of shared vehicles in urban areas.

Research Question 2: *Can we create an algorithm that can efficiently identify urban communities, by considering a set of factors?*

In order to address this question, we develop a multi-objective greedy optimization algorithm that is able to identify communities in cities by taking into account three different factors: *i)* patterns of shared mobility usage, *ii)* distance between areas, and *iii)* structural informationIn Chapter 3 and in Fiorini et al. [60], we demonstrate the effectiveness of our solution through a qualitative analysis of the results obtained in a real-world scenario (the city of Milan, Italy).

Research Question 3: *Can we develop and efficiently implement novel Deep Learning architectures that enhance the ability to extract information from spatio-temporal mobility data?*

To address this research question, we propose two novel architectural solutions, 3D Convolution LSTM on Spatio - Temporal (3D-CLoST) and Spatio Temporal REsidual Encoder-Decoder Network (STREED-Net), that incorporate innovative approaches to capture spatial and temporal dependencies within data. We present these models in Chapter 4 and in Fiorini et al. [59, 62], and we demonstrate that they have significant advantages over state-of-the-art methods in terms of predictive accuracy for the short-term flow forecasting.

Research Question 4: *Can spectral Graph Convolutional Networks solve mobility problems that involve the management of directed graphs with both positive and negative weights of arbitrary magnitude?*

In Chapter 5 and in Fiorini et al. [63], we introduce a new solution for spectral-based Graph Convolutional Network (GCN) using a new positive semidefinite Hermitian matrix called *Sign-Magnetic Laplacian*. This extends the use of spectral-based graph convolutional networks to directed graphs with non-restricted edge weights.

Research Question 5: *Can Deep Neural Networks and Statistical Models accurately predict the indices that describe the pattern of city parking?*

In Chapter 6, we conduct a study on parking patterns in the city of Rome. The work begins by identifying appropriate indexes for the analysis of parking. Then, through extensive experimentation, we compare

different prediction methods, among statistical models, graph convolutional networks, and Convolutional Neural Network (CNN)s, to evaluate the performance differences between various models.

2

Explaining Adoption Patterns

Recent years have witnessed the emerging of novel shared mobility solutions that provide diffused on-demand access to transportation. The widespread adoption of these solutions, particularly electric mopeds (e-mopeds), is expected to bring important benefits such as the reduction of noise and atmospheric pollution, and road congestion, with extensive repercussions on liveability and quality of life in urban areas. Currently, almost no effort has been devoted to exploring the adoption patterns of e-moped sharing services, therefore, optimal management and allocation of vehicles appears to be a problem for service managers. In this study, we tried to demonstrate the validity of the hypothesis that the adoption of electric mopeds depends on the built environment and demographic aspects of each neighbourhood. In detail, we singled out three features concerning the area characteristics (distance from centre, walkability, concentration of places) and one about the population (education index). The results obtained on a real world case study show the strong impact these factors have in determining the adoption of e-moped sharing services. Finally, an analysis was conducted on the possible role that the electric moped sharing can play in social equalization by studying the interactions between rich and poor neighbourhoods. The results of the analyses conducted indicate that communities within a city tend to aggregate by wealth and isolate themselves from one another (social isolation): very few interactions, in terms of trajectories, have been observed between the richest and poorest areas of the city under study. Research reported in this chapter has been published in [61].

2.1 Introduction

Understanding urban mobility patterns is becoming increasingly important for planners, administrators, and transport providers. Indeed, the growing concern for the environment, demographic changes, citizen lifestyle, and economic issues are constantly redefining urban mobility. Urban transport has a significant impact on the

quality of life of most people, and this is especially true today, as more than 50% of the world population lives in urban areas, with Europe reaching 75% [205]. Furthermore, by 2050, almost 68% [189] of the world population is predicted to live in urbanized areas, exacerbating human mobility and freight transport problems, which are crucial drivers for economic development and livability in the city [18]. As pointed out in [68], another important issue concerns air pollution, which entails great environmental risks for citizens. To overcome these challenges, institutions are enacting air quality plans and mobility strategies, in which transport alternatives to traditional mobility play a key role. In this sense, electric shared mobility can represent an opportunity for local administrations to rethink urban transport in terms of sustainability by disincentivizing bulky private motor vehicles, which often cruise the city with only the driver. The resulting reduction in traffic and air pollution would result in health benefits and cost savings for the citizens and a reduction in environmental impact on the community [138, 162].

In the field of shared vehicles, e-moped sharing is increasingly assuming a prominent position in Europe cities [84]: currently, 15 of the 22 countries, that have introduced the use of mopeds, are located in Europe and represent 54% of the total fleet deployed (in 2019 it was 58%). Germany (26), Poland (23), the Netherlands (19) and Spain (9) have the highest number of cities with moped-sharing services, with the Netherlands experiencing very strong growth over the past year. In the current situation, where many governments have signed the Paris Agreement [188] on climate change, electric mopeds can be a valid instrument to mitigate the pollution problem: on the one hand, city pollution can be directly reduced by the adoption of electric vehicles, and on the other, compact conveyances have a positive effect on the problem of road congestion, which is also a cause of pollution. In this sense, the authors of [206] show that the introduction of an adequate fleet of shared e-mopeds can replace a relevant percentage (up to 23%) of car trips. Therefore, e-mopeds represent a major enabler to build up a sustainable urban transport system and to improve city livability. Nonetheless, although several research contributions can be found in the field of shared mobility, mainly addressing car, bike, and e-scooter sharing services, almost no effort has been dedicated to studying the adoption of shared electric mopeds, despite its significant growth in recent years.

This chapter is organized as follows. After this introductory section, in Section 2.2 we present the variables used in the study with the related analysis methodology, while the relevant literature is reviewed in Section 2.3. In Section 2.4 we describe the case study and in Section 2.5 we present and discuss the main results of the study. Finally, Section 2.6 concludes the chapter and outlines future research.

2.2 Problem Statement and Methodology

Given a tessellation of the area of interest (i.e., the city) in regularly-shaped squares, called regions (equivalently, markers), the problem of adopting e-moped sharing, defined as the problem of predicting the shared use of e-mopeds and studying users' behavior within each region, has been little explored in literature. The few studies published on this subject contemplate the use of simulation [206], information directly linked to users (surveys) and mobility data [2, 3], but this knowledge is not always available and easy to access.

Our primary goal is to study the adoption of e-moped sharing services within the urban context by correlating utilization patterns to socio-economic and structural factors. Secondly, the study aims at finding evidence of the social isolation phenomenon by identifying meaningful social categories based on the economic status and calculating the rate of interaction among them. In what follows, the methodology implemented is outlined.

The study began with the definition of the target variables, which act as proxies for the adoption of e-moped sharing services. Once this first part was completed, we identified the dependent variables trying to meet two driving requirements: *i)* they must be easily accessible, and *ii)* they must show a close relationship with the

built environment and demographic aspects of the city. Finally, after applying a feature selection method, a linear regression was performed with the selected covariates. The results obtained were evaluated both in terms of the adjusted R-square and the importance of the independent variables.

It is worth pointing out that while mobility is characterized by well-known temporal dependencies (e-mopeds data presents trends and seasonality which depend on the day of the week and the time of day [59, 62]), the socio-economic and structural characteristics of the city are represented by covariates that can be considered almost constant in the medium term. Correlating those factors temporally is therefore impossible. To overcome this issue, we have identified in the literature some mobility indicators that are also measures of a central tendency [197]. A detailed description of the dependent and independent variables used in this study is given below.

2.2.1 Dependent and Independent Variables

In this work, the adoption of shared e-moped is evaluated through two different variables: the travel radius, which provides a sensible proxy for how much the service is being used, and the average daily flow, which focuses on how it is being used.

Travel radius (r_g). It is the average distance (in meters) travelled by the e-mopeds of the region under study i towards all the other markers to which the e-mopeds are directed [197]. The radius is calculated using Equation (2.1):

$$R_i = \sqrt{\frac{1}{n}\sum_{t=1}^{n}\Big[2g \times sin^{-1}\Big(\sqrt{sin^2\Big(\frac{\varphi_t - \varphi_i}{2}\Big) + cos\varphi_i cos\varphi_t sin^2\Big(\frac{\phi_t - \phi_i}{2}\Big)}\Big)\Big]^2} \tag{2.1}$$

where n is the total number of regions reached by e-mopeds starting from i, t identify a marker visited by e-mopeds starting from i, ϕ_t (ϕ_i) is the latitude, φ_t (φ_i) is the longitude of the marker t (i, respectively) and g is the radius of the earth in meters.

Average daily flow (a_f). It is the total e-moped flow measured in a region divided by the number of considered days, as shown in Equation (2.2):

$$AF_i = \frac{\sum_{n=1}^{N}(\iota_{n,i} + \omega_{n,i})}{N} \tag{2.2}$$

where N is the total number of analyzed days, $\iota_{n,i}$ is the total flow of day n entered marker i and $\omega_{n,i}$ is the total flow of day n leaving marker i.

As mentioned before, three independent variables were selected that carry information about the characteristics of the region (distance from center, walkability and concentration of places), and one feature to characterize the population (education index).

Distance from center (x_d). It is the distance in meters between the center of a marker and the point considered the geographic center of the city (it accounts for decentralization). It is measured by applying the Haversine formula [150], which determines the shortest distance between two points on the surface of a sphere (in this case the earth).

Education index (x_g). It is the indicator that measures the share of graduates, i.e., the ratio between the number of graduates and the total number of residents, in each region of the city. It is used as a proxy for the average socio-economic status of an area, as in [15, 27, 144] it was highlighted how higher education is related to higher wages and greater economic and social well-being. Specifically, in [144], the authors point out that higher education includes higher lifetime earnings, a more satisfying work environment, better health, longer life, more informed shopping, and a lower likelihood of unemployment.

The choice of selecting this indicator was made due to the lack of detailed salary information (for the case study presented in Section 2.4) for privacy reasons. The education index is calculated considering the population group aged between 25 and 70. The choice of this demographic cohort should shield the analysis from possible bias due to the tendency of off-site students to live near universities and colleges; nonetheless, any information about young people domiciled but not residing in the city analyzed in this study was removed.

Walkability (x_s). It was proposed in [24] and is the index that measures the city's street network orientation distribution. In particular, streets' orientation is the angle that the vector, joining its start to end coordinates, makes with North. In other words, the streets' orientation entropy measures the variability in their respective azimuths. It is calculated as shown in Equation (2.3):

$$H_i = -\sum_{k=1}^{n} P(o_k) \log P(o_k) \tag{2.3}$$

where n represents the total number of bins, k indexes the bins, and $P(o_k)$ represents the proportion of orientations that fall in the k^{th} bin. In more detail, Boeing [24] proposes a measure for disorder in city street orientation (H_i), whereas, two follow-up studies [41, 42] have used the measure H_i to link it to walkability and physical activity. Specifically, [42] operationalized the measure from [24] using the entropy to capture city imageability/legibility, and [41] used the same entropy measure to show that street entropy is strongly associated with physical activity. Based on these studies, a high value of H_i corresponds to a disordered region in terms of the road graph, which encourages walking. On the other hand, a low value of H_i indicates that the region has more orderly road networks, which are less conducive to walking and require more driving or transportation.

Concentration of places (x_p). It measures the diversification of Points of Interest (POIs), P_i, through the calculation of entropy with respect to the different classes of POIs present in the macro-categories of services and buildings, such as entertainment, commercial, transport and healthcare. Concentration of places is calculated following Equation (2.4):

$$P_i = \frac{-\sum_u p_{u,i} \log(p_{u,i})}{\log(|P_i|)} \tag{2.4}$$

where $p_{u,i}$ is the proportion of the POI category u in zone i and P_i is the set of POI categories in i. The numerator in this expression is the Shannon entropy [118] associated with the different categories of POI in i, normalized by the log of the number of categories of POIs present in i.

Regions that have an elevated diversification of POIs with equal propensity have high entropy, while areas that tend to have numerous POIs of few categories are considered to have low entropy.

2.2.2 Linear Regression Model

To predict the adoption of electric mopeds and measure the effect of the different covariates, inspired by the approach presented in [184], a regression model was used as it is a reliable method to quantify the impact of each variable on a topic of interest [51]. Therefore, given the feature vectors $x_d \in R^n$ and the target value $y_d \in R$, we aimed at predicting $\hat{y}_d$ using an Ordinary Least Squares regression model of the form:

$$\hat{y}_d = w_o + w_1 x_d^1 + ... + w_n x_d^n \qquad d = 1, ..., N \tag{2.5}$$

where the coefficients w_j are learned by the model, n is the number of predictors and N is the total number of observations.

The results of the regression model were evaluated with the Adjusted R-squared, which is a modified version of R-squared that is adjusted for the number of predictors in the model. It is used to determine how reliable the correlation is and how much it is determined by adding independent variables. In terms of values, it increases when a new term improves the model more than would be expected by chance, while it decreases when a predictor improves the model less than expected [128].

Finally, a feature selection was applied to remove non-informative or redundant predictors from the model [105]. In detail, the best combination of independent variables was identified in terms of Adjusted R-squared.

2.2.3 Well-off Groups and Interaction Ratio

In order to divide the different markers into well-off bands, we used the education index as a proxy for a socio-economic indicator. More in detail, we calculated the quartiles of the education index distribution and split the markers into four different groups, viz., *very deprecated* (VD), *deprecated* (D), *well-off* (W) and *very well-off* (VW). Let $\mathscr{C} = \{VD, D, W, VW\}$ be the set of the four classes, the interaction ratio $Ir_{CC'}$ between two classes C, C' can be calculated following Equation (2.6):

$$Ir_{CC'} = \frac{\sum_{w \in C} \sum_{d \in C'} x_{wd}}{\sum_{P \in \mathscr{C}} \sum_{w \in C} \sum_{d \in P} x_{wd}} \qquad \forall (C, C') \in \mathscr{C}^2 \tag{2.6}$$

where x_{wd} is the number of interactions (trips) observed between marker w and marker d. Therefore, the interaction ratio represents the percentage of observed interactions between markers belonging to two socio-economic classes and approximates the probability of a trip beginning in the markers of one class and ending in those of the other.

2.3 Related Prior Work

Over the past few years, the attention towards shared mobility services has grown steadily, generating worldwide interest. This has led to considering shared mobility as one of the three revolutions of urban transport along with electrification and automation of vehicles [65]. The meaning of the phrase *shared mobility* is twofold; on the one hand, it refers to a service that provides vehicles to be shared between different users with a pay-per-use billing model. Examples of those services are: car sharing, bike sharing, or scooter sharing (both mopeds and kick-style). On the other hand, it may refer as well to scenarios in which a single ride is shared [170]. In this study, we consider the first characterization, where the user is provided with *as-a-service* short-term access to a mean of transport.

Shared mobility has a strong impact on urban areas, as it is often considered a valid instrument to address the problems of traffic congestion and air pollution. Among the benefits of shared-use vehicle systems, in [14] the authors highlighted four main advantages with important consequences on the quality of life: *i)* provision of mobility alternatives that are more flexible than public transport and cheaper than the private vehicle, *ii)* potential to lower human transport costs and reduce the need for city parking areas, *iii)* improvement of air quality since most solutions are based on electric or hybrid-electric vehicles and *iv)* access to and encouragement of the use of more efficient and environmentally friendly modes of transport.

The early forms of alternative mobility came in the shape of bike-sharing services, followed by car-sharing schemes. Later, moped sharing has established itself as an innovative mobility model in urban areas, with some peculiarities that make it an attractive option, especially for short distances in the central areas of the city. Finally, in recent years, electric scooters have also been introduced with great success. Existing literature on shared mobility focuses mainly on car sharing or bike sharing. Regarding car sharing, previous studies have widely analyzed the typical profile of users [170] or its impact on modality competition [44]. While, in the bike sharing field, many authors have extensively explored the main factors for and barriers to the adoption of bike sharing systems [43], as well as its effect on urban congestion [195]. Finally, in the last few years, research focused on the benefits and use of electric scooters has also been growing [21, 10].

To the best of our knowledge, e-moped adoption is poorly understood, despite its significant growth in recent years. There are only a handful of works dealing specifically with mopeds, also due to the limited data available. In fact, [2] and [3] grounds their analyses solely on an online survey conducted in Spain, [46] and [143] use actual mobility data; lastly, [206] uses a simulator to generate synthetic mobility data. More in detail, in [46] a first approach was tested for the clustering of users of mopeds. The authors developed a cluster analysis to study moped-sharing users and identify customer segmentation based on scooter-sharing usage data. Users are profiled according to four variables: *i)* age, *ii)* time between rides, *iii)* distance driven, and *iv)* revenue per customer. In [2] the authors focused on identifying the characteristics that most influence the use of electric moped sharing. They adopted a generalized ordered logit model to explore the key factors determining the use of shared e-mopeds. In [143] the authors used GIS and GPS data to identify optimal locations for moped sharing parking spaces in central Madrid. A first overview of the moped sharing demand by exploring the usage and views towards this new mobility alternative is provided in [3]. To this end, they conducted Kruskal-Wallis tests to identify the segment of the urban population that is most likely to adopt moped sharing, and further statistical mean differences were made in specific variables related to moped sharing. Finally, [206] investigates the capacity of a shared platform of electric mopeds to replace the transport by passenger cars in Berlin by developing a simulation-based analysis. Based on the data generated by a simulator, the authors also provide holistic environmental and economic views through an analysis of the impact on the life cycle of the fleet.

In conclusion, the literature analysis suggests that this study is the first to analyze in detail socio-economic and structural factors behind the utilization patterns of e-mopeds in a city. In addition, it also explored the interplay between e-moped use and social isolation of the more and less affluent social classes within the city.

2.4 Case Study

A real word case study was considered for the experimental analysis: **E-Moped Sharing Dataset** contains records of vehicle pick-ups and drop-offs over one week (December 2-8, 2019) for the city of Milan, Italy. The coordinates of each vehicle, the sampling time, and the reference marker, i.e., the one the vehicle falls within, are provided. Sampling has been performed at regular intervals of 15 minutes. By analyzing the dataset, it is possible to trace the trips made by the vehicles and construct the transition matrix containing the probabilities

of displacement. In this case, the central section of Milan has been divided into 223 regular markers with sides of 400 meters, as shown in Figure 2.1. The marker size, set in the original dataset, is comparable with those used in other studies [2, 60, 206]; it can reasonably be considered adequate as corresponds to a distance easily walkable to reach a vehicle. The dataset was provided to us by FLUCTUO, a European leading aggregator of data on shared mobility services, for research purposes.

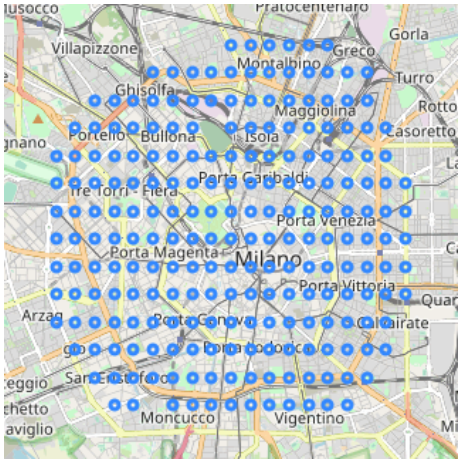


Figure 2.1 Area of analysis

2.4.1 Dependent and Independent Variables

Travel radius. Figure 2.2 shows the travel radius distribution. As can be seen, the travel radius is directly proportional to the distance from the city center; indeed, there is a positive correlation (0.57) between r_g and x_d. This means that people from areas further from the city center tend to travel on average longer than those who start trips from the center. This is easily explained considering that in the historical center of the city of Milan are concentrated the main economic, social, and shopping districts.

Figure 2.2 Travel Radius

Average daily flow. This target variable was calculated by applying the Haversine formula [150]; however, it was decided to set a lower bound on the number of trips to be considered. In particular, the connections

between regions that registered only one transfer for the whole week were not taken into consideration. In this way, unusual movements were eliminated.

Distance from center. The city of Milan (Ita) has a street network conformed of concentric circles deriving from its medieval legacy. Moreover, as for many European cities, the cathedral (Duomo), with coordinates (45.464211, 9.191383), can still be identified as the geographical (as well as, cultural and economic) center of the city. Admittedly, the study of this metric requires the prior identification of a point to be considered the city center, intended as an attraction and aggregation pole such as to have a substantial impact on mobility, and this is not always straightforward as in the case of the city of Milan. In these cases, knowledge of the city and careful data analysis are required. In addition, a modern city might be polycentric with distinct economic and social centers. In these cases, alternative approaches must be considered; for instance, the distance from the closest *center* could be used.

Education index. The data used to construct the education index were obtained from the ISTAT (The Italian National Institute of Statistics) portal [89] relating to the 2011 census. As shown in Figure 2.3, the percentage of graduates is higher in the more central areas of the city, while it tends to decrease moving radially towards the suburbs. Furthermore, the strong negative correlation (-0.74) between x_g and x_d highlights an inverse proportionality between the two variables. This result confirms the soundness of considering the percentage of graduates as a proxy feature of the socio-economic status of an area.

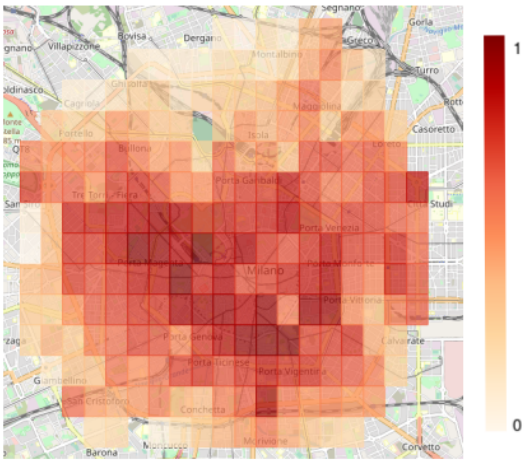


Figure 2.3 Distribution of the proportion of graduates in the city

Walkability. Figure 2.4 shows the distribution of Walkability in Milan. The index values in the image tend to be high, especially in the central part of the city and near the green areas. This result can be explained by the highly pedestrian nature of the center of Milan where several areas are inaccessible to cars.

Concentration of places. The distribution of the variable is asymmetric, pointing towards values between 0.7 and 0.8 with mean 0.75 and standard deviation 0.14. Such a distribution strongly depends on the portion of the city considered in this work, as being the central area of the city, the concentration of POI is quite homogeneous with only a subtle difference between the different markers.

Correlation between independent variables. We computed the correlation coefficients between all the independent variables, as shown in Table 2.1, and found that there is a strong negative correlation (-0.74) between Distance from center and Education index. This could be a sign of collinearity among the predictors.

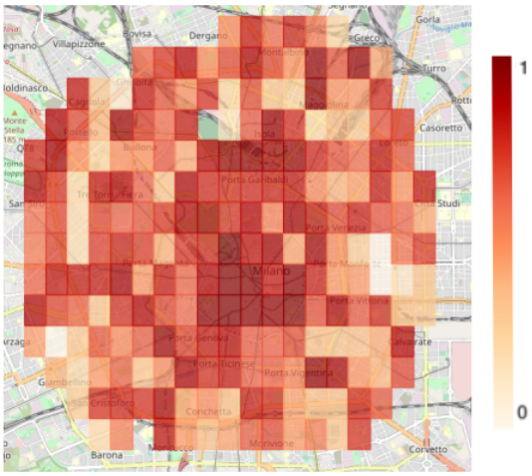


Figure 2.4 Walkability index in Milan

To further asses this, we calculated the variance inflation factor among the independent variables, and indeed confirmed that the Distance from center and Education index have higher VIF values Table 2.2. This could be indicative of the fact that higher education may afford people to move to larger suburban houses. This phenomenon needs to be accounted for in any further analysis in the form of an interaction variable between education and distance from center.

Table 2.1 Correlation between the independent variables

Independent variables	Correlation			
	Distance from center x_d	Education index x_g	Concentration of places x_p	Walkability x_s
Distance from center x_d	1	-0.74	0.23	-0.21
Education index x_g	-0.74	1	-0.15	0.20
Concentration of places x_p	0.23	-0.15	1	-0.18
Walkability x_s	-0.21	0.20	-0.18	1

Table 2.2 Variance Inflation Factor

Variables	VIF
Distance from center x_d	2.25
Education index x_g	2.19
Concentration of places x_p	1.077
Walkability x_s	1.072

2.5 Experimental Analysis

In this section, we initially present the analyses conducted on the dataset for its validation, and subsequently, we discuss the results obtained from the experiments carried out to predict the two target variables: travel radius and average daily flow.

2.5.1 Dataset Validation

To perform the experiments, the 223 squares were aggregated in *i)* districts and *ii)* municipalities. Districts are census areas, which consist of neighboring blocks (i.e., sections delimited by street segments) grouped based on socio-economic conditions [132], while Municipalities are a different subdivision of the territory decided starting from 1999.The subdivision implies that each Municipality, except for the central area, extends from the semi-central area to the periphery and acquires partial administrative autonomy.

Therefore, as previously done in [159], each marker was assigned to a specific district. If an area overlapped several districts, as can be seen in Figure 2.5a, it was assigned to the district with which it had the largest overlap. We performed the same procedure for the assignment of squares to municipalities, since, as shown in Figure 2.5b, a similar scenario occurs. In the end, 46 districts were analyzed, about half of the districts in the entire city, and all nine municipalities.

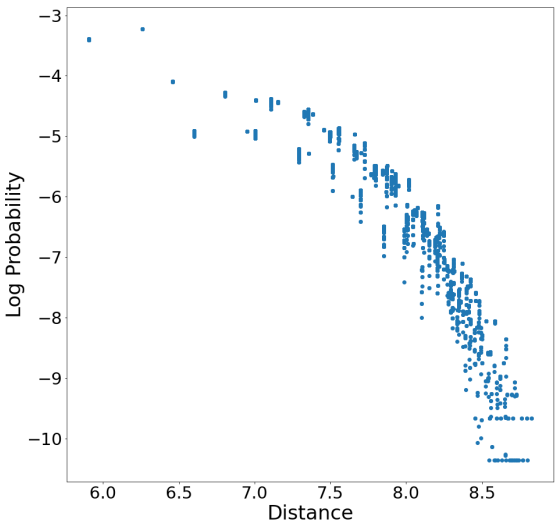
One of the issues that can arise from the aggregation of spatial data is the Modifiable Areal Unit Problem (MAUP), i.e., an effect of statistical bias appearing when samples in a given area are used to estimate aggregated information. For this reason, we decided to perform two different aggregations (namely, at the municipality and district level) and in the analyses we carried out, we compared the behavior of the variables analyzed in both scenarios to check the consistency of the results obtained.

(a) District subdivision

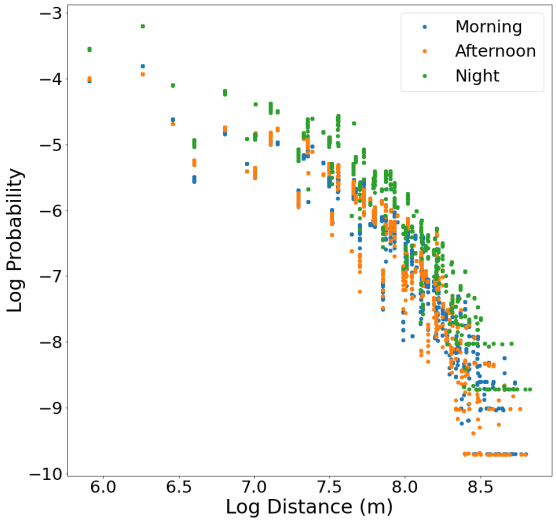
(b) Municipalities subdivision

Figure 2.5 (a) District subdivision; (b) Municipalities subdivision

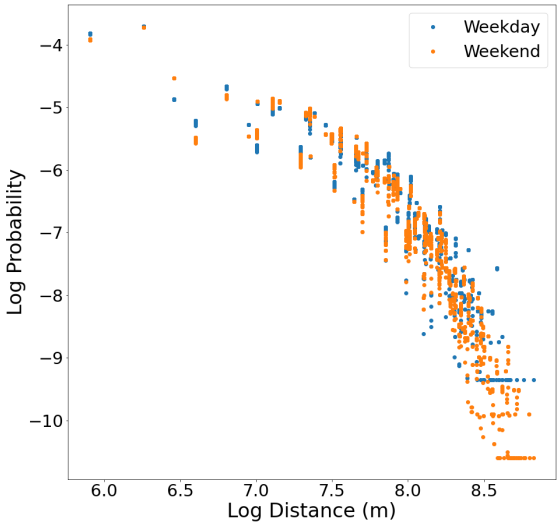
Finally, it was tested whether *E-Moped Sharing Dataset* verified the hypothesis that the distance frequency distribution is exponentially distributed; this derives by the assumption that people rarely move away from familiar areas, traveling to a limited number of nearby locations, and therefore short-range movements are more frequent than long-range ones. The data on e-mopeds in Milan validate the hypothesis: the Figure 2.6a shows the probability (log) of traveling a certain distance, and as can be seen from the image, as the distance increases, the probability of traveling decreases. Furthermore, it was investigated whether the considered dataset remained consistent with the geographic closeness hypothesis when different temporal aggregations are considered. More in detail, the original records were aggregated by daily time slots (morning, afternoon and evening) and by days of the week (Weekdays and Weekends). As shown in Figure 2.6b and Figure 2.6c, the pattern is also evident at these levels. It can also be seen that trips made at weekends and in the evening have a higher probability of making long distances. This evidence confirms what has been shown in [3], which points



(a) Log probability of one's travelling a certain distance



(b) Log probability of one's travelling a certain distance across different time slots



(c) Log probability of one's travelling a certain distance across weekdays and weekend

Figure 2.6 Log probability of one's travelling

out that electric mopeds are widely used for leisure activities, as opposed to free car sharing where the main use is for commuting and airport transfers [16].

It should be noted that in all three graphs, trips over 5 km have a very low probability. This behavior is consistent with what has been observed in the literature: in [206], for instance, the authors observe that the average distance traveled with an electric moped in Berlin is between 3.6 and 4.1 km.

2.5.2 Travel Radius Prediction

Table 2.3 shows the results of the two regressions constructed to predict the travel radius. Analyzing in detail the regression related to the district aggregation, one can see that three out of the four variables (x_d, interaction

Table 2.3 Radius prediction

	Radius	
	Districts	Municipalities
Distance from center x_d	1781.75^{***}	-
	(505.21)	
Education index x_g	580.18	-379.25^{**}
	(421.22)	(73.96)
Interaction of x_d and x_g	-1661.34^{**}	-
	(730.81)	
Walkability x_s	-399.83^{**}	-421.22^{*}
	(129.61)	(132.93)
Concentration of places x_p	-	-
Constant	2886.31^{***}	3880.88^{***}
	(496.36)	(344.22)
Observations	42	9
R-squared	0.569	0.922
Adj. R-squared	0.522	0.896

$^{*}\ p < 0.05$, $^{**}\ p < 0.01$, $^{***}\ p < 0.001$
Note: Robust standard errors in parentheses

of x_d and x_g, and x_s) are significant for the prediction of the radius. The four covariates jointly explain the variance of 56.9%, and we get a ${R_{Adj}}^2$ of 0.522.

The significance of the variable x_d indicates that, as the distance from the city center of an area increases, the length of the travel radius also increases: the distances that users, who are in more peripheral areas, have to travel to reach different places in the city are greater than those traveled in trips that start in the central area of the city. Therefore, users in areas with vehicle availability but in a less central position use moped sharing to travel longer distances than users in the city center. This result is in line with what has been observed in studies on bike sharing, where transfer distances are generally longer in suburban areas where public transport services are typically less available [113]. The significance of the variable x_s explains how the use of mopeds is also negatively influenced by the walkability: pedestrian areas have a reduction in the distance traveled. Therefore, these are areas where the environment is suitable for walking and does not encourage the use of e-mopeds.

Regarding the results obtained from the aggregation in municipalities, the value of R^2_{Adj} cannot be trusted due to the small sample size (9 observations). Nonetheless, in the following, the behavior of the covariates is discussed and compared against that seen in the districts scenario in order to check the possible presence of MAUP. It emerges how the variable x_s is significant, and its behavior is the same as seen previously. Furthermore, x_g is also negatively related to the target variable: as the education index increases, the travel radius decreases. This behavior is well explained by the fact that the richest zones are in areas of the city where shopping and leisure districts are located. Incidentally, those zones are also better covered by public transportation and feature a higher degree of walkability, making long-distance travels less frequent.

2.5.3 Average Daily Flow Prediction

Table 2.4 shows the results obtained from the regression regarding the daily adoption of shared scooters.

Table 2.4 Average daily flow regression

	Average daily flow	
	Districts	Municipalities
Distance from center x_d	12.06^{***}	2.32^{*}
	(2.62)	(0.77)
Education index x_g	16.97^{**}	-
	(5.53)	
Interaction of x_d and x_g	-11.60^{***}	-
	(7.97)	
Walkability x_s	-	-
Concentration of places x_p	-1.42	-6.12^{*}
	(1.23)	(2.04)
Constant	-4.01	5.46
	(2.16)	(1.60)
Observations	34	9
R-squared	0.580	0.756
Adj. R-squared	0.522	0.675

$^{*}\ p < 0.05, \quad ^{**}\ p < 0.01, \quad ^{***}\ p < 0.001$
Note: Robust standard errors in parentheses

Analyzing the result obtained with the aggregation in districts, the four covariates explain 58% of the target variable, with a ${R_{Adj}}^2$ of 0.522. The significant predictors are x_d (p-value ≤ 0.001), x_g (p-value < 0.01) and the interaction between x_d and x_g (p-value ≤ 0.001). We set a lower bound, as explained in section 2.4, and we had a reduction of 19% in the number of observations. Table 2.4 shows the significance of the variables x_d and x_g. It can be seen with x_d that moving away from the center leads to an increase in daily scooter use. Similarly, an increase in the education index x_g leads to an increase in scooter use. The latter behavior has also been observed in bike sharing, where variables such as *educated*, *working*, *high income* are strongly correlated to the user profile of bike sharing services [52]. Also in the case of car sharing, people with graduate degrees are more likely to use this service; nonetheless, unlike what we see for e-mopeds, living in city center seems to encourage the use of car sharing [16].

Even if the variable is not statistically significant, an increase in x_p leads to a reduction in the daily flow: since as the entropy grows, the diversification of the available POIs increases, this effect is synonymous with a well-supplied area and from which it is not compulsory to move away daily.

Finally, also for this prediction, the regression carried out with the aggregated observations in municipalities cannot be evaluated in terms of R^2_{Adj} for the small number of observations. Again, to check for MAUP, we verify the consistency of results in the municipalities and districts scenarios.

Interestingly, after feature selection, the same two covariates, x_d and x_p, remain, and both are significant (p-value ≤ 0.05). The first one turns out to be significant also in the district scenario, while the second one is

not. Nonetheless, the parameters associated with x_d and x_p have the same signs in both scenarios, suggesting that the presence of MAUP can be ruled out.

2.5.4 E-moped Sharing is not a Social Equalizer

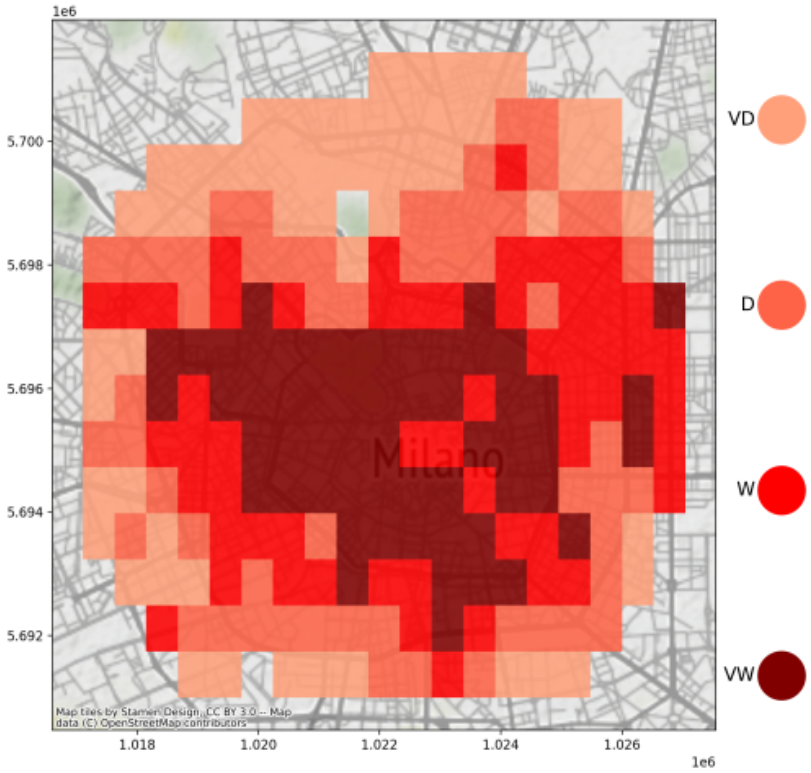


Figure 2.7 The four different communities based on the well-off indicator shown on the city map

In recent years, several pieces of research have focused on social isolation in cities, where it has emerged that social isolation is experienced by residents of highly disadvantaged and highly advantaged neighborhoods because the two groups spend time in largely non-overlapping parts of the city [104]. Additionally, a surprisingly high consistency was identified between neighborhoods of different races and income characteristics in average walking distances (in meters) and the number of unique neighborhoods visited in the metropolitan region [152].

We are interested in analyzing the phenomenon of social isolation from the point of view of shared moped users. Our goal, in fact, is to verify whether mopeds can be considered vehicles of social equalization. Therefore, we initially identified four different communities, shown in Figure 2.7 for the city of Milan, based on the well-off indicator: *i) very deprecated* (VD), *ii) deprecated* (D), *iii) well-off* (W) and *iv) very well-off* (VW); then, we calculated the interaction ratio reported in Figure 2.8 (see subsection 2.2.3 for more details). As can be seen from the image, the iteration ratio between areas of the same community is strong, but it is interesting to analyze the behavior of users of the more affluent and poorer areas: there are few interactions between these two communities. In detail, the interaction ratio of wealthy areas (Community VW) with less well-off areas (Community VD) is 0.08%, while on the other direction is 0.17%. Observing the tendency of the two curves from Figure 2.9, an evident trend emerges: the greater the distance in terms of well-being between communities, the lesser the interaction.

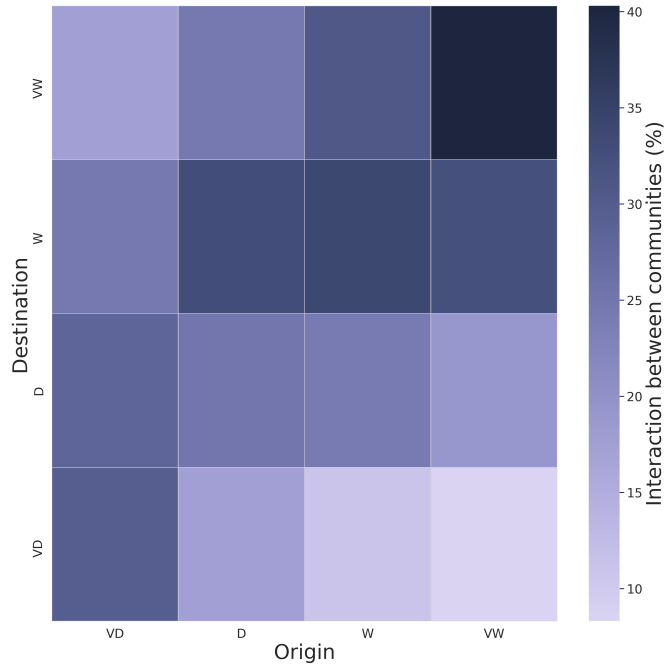


Figure 2.8 Interaction ratio between communities

In order to double-check the consistency of the analyses, we observed the behavior of two markers belonging to the city center, an area where the education index is generally high; both belong to the VW class. For both the ratio of interaction with markers of the same social level is very high (32% and 37%). The interaction ratio decreases dramatically when we consider the movements between these areas and the poorer areas of the city. In this case, the interaction ratio falls to 10% and 7% respectively. A similar behavior was observed when we picked two markers with a low value of education index, one from the north of the city and one in the west. Also in this case, the interaction ratio with markers of the same social level is high (31% and 27%, respectively). While, the interaction ratio with markers for the most well-off areas, the ratio decreases to 14% for the first marker and 16% for the second.

The results obtained are consistent with known dynamics of large cities, in which communities tend to be homogeneous in terms of wealth, isolating themselves (even geographically) from one another. In this scenario, even an agile mobility mean, such as the electric moped, does not seem to play an important role in reducing social isolation.

2.6 Conclusion

The identification of the variables that allow to better explain and understand the adoption of ecological alternative mobility solutions (like shared e-moped) seems to be a topic not yet explored. Besides the relative novelty of these services, it is believed that the main cause of this void in the literature is due to the difficulty of

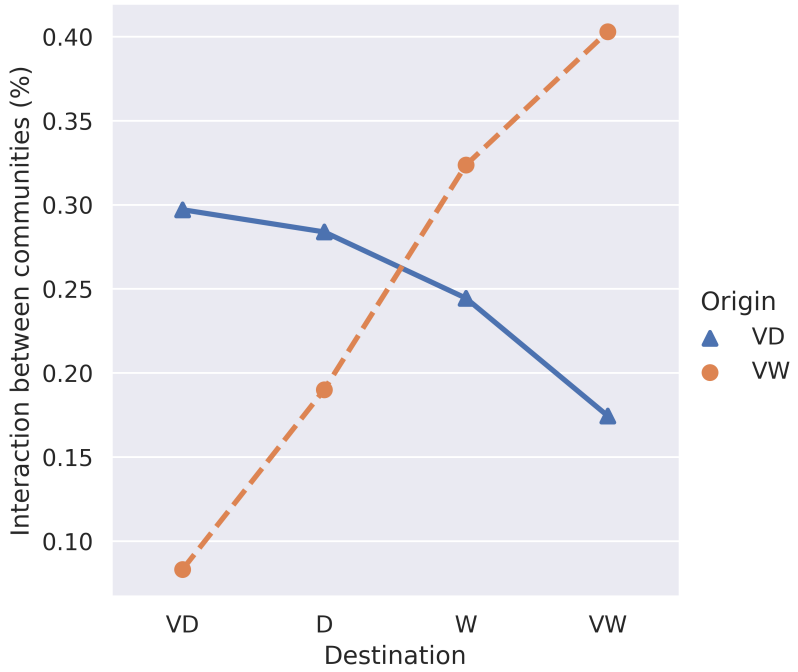


Figure 2.9 Trend of community VD and community VW interactions

acquiring data to study. The handful of works on this topic in fact make use of surveys administered directly to service users or simulation tools.

This chapter presents a study that analyses four different variables related to territorial and population characteristics without using any direct information on users. The analysis allowed us to explore the sharing behavior of e-moped in the central area of the Italian city of Milan, where users experience limited problems in terms of service delivery. Compared to past studies that limited themselves to highlighting a greater use of e-mopeds in the city center, our study suggests that, under the assumption that there is no shortage in the supply of vehicles, the use of shared e-mopeds (average daily flow) and the distance traveled (travel radius) is directly proportional to the distance from the city center. Furthermore, the impact of two of those variables (namely, concentration of places and walkability) on the adoption of the sharing moped seems to suggest that the diversification of the POIs and the organization of the roads play a central role in explaining mobility patterns. Therefore, urban and traffic planners who deploy sharing services could use the outcomes of this work as support for the distribution and allocation of vehicles in cities, especially when available mobility data for the service is limited or missing. Finally, we analyzed the possibility that the e-moped sharing service could play a role in social equalization, studying the interaction ratio between neighborhoods according to their socio-economic status; unfortunately, the results of the analyses conducted indicate that communities within a city tend to aggregate by wealth and isolate themselves from one another. Very few interactions, in terms of trajectories, have been observed between the richest and poorest areas of the city.

Limitations and Future Work. The biggest limitation of this study concerns the limited dimensionality of the dataset both in terms of time horizon (only one week of data) and in terms of space (Milan's suburbs are excluded). Nevertheless, the results obtained are statistically significant and consistent with well-known dynamics in large cities. Future developments will involve studying the impact of the considered variables on

other use cases (different sharing services and cities) to uncover similarities and differences with respect to their predictive power on the particular sharing service adoption.

3

Area Clustering

Characterizing urban communities is essential for understanding citizens' needs and neighborhood-wise dynamics. Discriminating factors are population mobility patterns, neighborhood structural characteristics, and distance to other areas of the city. Available approaches focus on one aspect and, often, suffer from isolated nodes and excessive geographical fragmentation of solutions. For these reasons, we formulate the problem of urban community clustering considering all three aspects and provide an algorithm that combines hierarchical aggregation with node adjustment and relocation. We evaluate our approach on a real-world data set and the obtained results show its efficacy. Finally, we also show the importance of using map embedding for characterizing neighborhood from the structural standpoint. Research reported in this chapter has been published in [60].

3.1 Introduction

The continuous evolution of urban mobility, which involves citizens' commuting behavior, and the frenetic structural transformations that occur within the city are making the municipality's zoning obsolete. Indeed, there is less and less evidence of homogeneity within these (often artificial) urban communities; in this context, the phrase *urban community* refers to a contiguous geographic region with dense crowd aggregation and homogeneous characteristics [112].

The problem of identifying the structure of the city communities is becoming increasingly important, especially from the point of view of policymakers, who would have a tool for knowing which areas show similar characteristics. This problem features strong topological, temporal and context components. In fact, distances and the city street layout (topological component), and the morphological characteristics of each neighborhood (context component or semantic information) have a profound impact on mobility over time with evident seasonal patterns (temporal component). In this study, the geographical space of interest (city) is logically partitioned into a regular grid of dimension $N \times M$ oriented by longitude and latitude. Each element

of the grid (*region*) is addressable through a pair of (n,m) coordinates corresponding to the n^{th} row and the m^{th} column of the grid.
According to [120], there are currently two challenges in the field of urban community structure identification:

- **Mixed Objective:** Identifying urban community structure with mixed objectives is a difficult NP-problem [221], and even designing scalable and accurate heuristics is an arduous task, especially on complex graphs.
- **Structural Complexity:** Potential communities are difficult to identify due to the inherent complexity of mobility and city structure. Frequently, isolated nodes and geographical fragmentation bedevil available solutions. This phenomenon is a consequence of the difficulty to identify a pattern in the sharing mobility data.

To address the first challenge, we propose a framework that considers three different aspects for the identification of urban community structures. We also addressed the structural complexity issue, using a combination of hierarchical aggregation and node-level adjustment to effectively capture the clustering structure.

The rest of this Chapter is organized as follows. In Section 3.2 we define the urban community structure problem and the proposed framework is described in detail. In Section 3.3, the literature on techniques used in location embedding and community identification is analyzed. In Section 3.4 data and results of experiments are presented and analyzed. Finally, the conclusions and recommendations for future work are discussed in Section 3.5.

3.2 Problem Statement and Proposed Framework

Given a partition of the area of interest (henceforth referred to as *city*) into regions of regular shape, the problem dealt in this work encompasses the identifications and clustering of similar areas in homogeneous neighborhoods, that is sharing similar characteristics in terms of structure (streets, buildings, parks, etc.) and human behavior (e.g. similar mobility patterns).

More formally, the problem can be seen as a clustering problem on a graph. In detail, a graph $G(V,E,A_v,A_e)$ with geographical zones, or markers, as nodes V, the edges E represent a route between two zones recorded by at least one mobility service, the distance among nodes and the transition probability associated with mobility information are attributes of the edges A_e and structural information of the regions as node attributes A_v.

Clearly, the ability of a graph to represent the static and structural characteristics of the city areas (with a major impact on the effectiveness of any clustering algorithm) depends heavily on what information is actually used and how it is represented as node attributes. This is an open problem that has been addressed in various ways in the literature. In the next section we describe an approach, based on Neural Networks, that can synthesize such physical features into vectors of real numbers.

In this section, we first show how discrete metadata describing the structural information of each area (e.g., number of streets) can be compressed and encoded as real-valued vectors; then, we present a clustering algorithm for identifying urban communities exploiting such a representation.

3.2.1 Encoding Structural Information

To identify urban communities, we consider structural information about each region. However, this information can be exploited following two different representations:

(a) (b)

(c) (d)

Figure 3.1 Map tiles

- **Metadata.** The clustering algorithm can use directly a heterogeneous vector of features where each element quantifies a specific characteristic such as the number of streets, the presence of parks and buildings.
- **Map Embedding.** Starting from an image of the map, referred to as *map tile*, the information is extracted and encoded as a vector of real numbers from a hidden layer of a neural network.

In this case, we have used the second methodology, which is similar to that proposed in [200]. The decision to use this methodology stems from the results of an analysis we carried out, in which it has emerged that the use of map embeddings is more effective in characterizing urban areas, as opposed to pure metadata. The idea behind this technique is to use two different types of data: *i)* map tile, the image, that describes the map in its geospatial information such as shapes and colors, and *ii)* metadata that quantify it, such as the number of streets, parks, buildings, etc. Although these two data are different in nature, they manage to describe the same area from different points of view and present a strong correlation. Therefore, the images are used as input and metadata as labels of a convolutional multi-label neural network based on ResNet50 [75], chosen for its proven efficiency in image processing tasks. The complete architecture of ResNet50 is used, excluding the last two layers, which are substituted by problem-specific layers.The first layer added is a fully-connected layer of size $512 \times E$ with activation function *SeLu*, where E is the size of the vector space. The model is completed by another fully-connected layer of dimension $E \times C$, where C is the number of target classes. The activation function is the *Sigmoid*, which is used to detect whether or not each of the labels of C exists for a given map tile. The vector (embedding) used as a node attribute is eventually extracted from the penultimate layer of the Neural Network, and the operation described is called Map Embedding.

Figure 3.1 shows two map tiles pairs. (a,b) has a cosine similarity of 0.98 (calculated using map embeddings) and 0.55 considering the metadata. This pair clearly shows a similar structure of the two regions; as a matter of fact, similarities are found in the presence of small parks, a similar distribution of transport stations, and the presence of a main street in both areas. Therefore, it appears evident that the comparison using map embedding is more effective than using vectors of metadata.

The pair (c,d) instead highlights two very different regions: the image on the left represents a section of the railway station of "Milano Porta Garibaldi" and it is characterized by a high presence of railway lines, while the image to which it is compared is an urban district. In this case, the similarity value calculated with map embeddings is 0.27, as opposed to 0.70 obtained using metadata.

3.2.2 Algorithm

The method we propose to identify the structure of the urban community exploits the combination of two complementary ideas: an initial hierarchical aggregation, considering each node as a community in its own right, and a subsequent adjustment and relocation of the nodes. This is a multi-objective greedy optimization algorithm designed to partition the graph G such that:

$$C^{\star} \leftarrow \underset{C}{argmax}(\delta Q + \beta I_d + \gamma E_d) \tag{3.1}$$

where C is the set of all possible clusters of urban areas, Q, I_d and E_d are the *Modularity* (Equation (3.2)), the *Inverse Distance* (Equation (3.4)) and the *Embedding base Distance* (Equation (3.6)) respectively, and δ, β and γ are non-negative real numbers. Note that the dependence of the objective function on C is not explicitly stated, so as not to lighten the notation.

Modularity. The problem can be seen as a clustering problem on a graph. This different point of view allows us to use modularity to measure the degree of connection among nodes. We compute the metric both *intra-cluster*, where the value is the resultant of all possible pairs of markers (v_i, v_j) with i and j belonging to the same cluster, and *inter-cluster*, whose value is the resultant of all possible pairs of markers (v_i, v_j) with i and j indicating two different clusters as follows:

$$Q = \frac{1}{2|E|} \sum_{v_i, v_j \varepsilon V} \left(w(e_{v_i,v_j}) - \frac{d(v_i)d(v_j)}{2|E|} \right) \tag{3.2}$$

$$\Delta Q = \frac{1}{2|E|} \sum_{i,j}^{c_{v_i}=C_X, c_{v_j}=C_Y} \left(w(e_{v_i,v_j}) - \frac{d(v_i)d(v_j)}{2|E|} \right) \tag{3.3}$$

where $w(e_{v_i,v_j})$ is the weight of the edge e_{v_i,v_j}, $d(v_i)$ is the degree of vertex v_i and $|E|$ is the number of the edges.

Inverse Distance. This metric accounts for the geographical distance among nodes in the same cluster *intra-cluster* and among cluster *inter-cluster*:

$$I_d = \frac{2}{|V|^2 - |V|} \sum_{v_i, v_j \varepsilon V} \left(\frac{1}{1 + \text{distance}(v_i, v_j)} \right) \tag{3.4}$$

$$\Delta I_d = \frac{1}{|E||X||Y|} \sum_{i,j}^{c_{v_i}=C_X, c_{v_j}=C_Y} \left(\frac{1}{1 + \text{distance}(v_i, v_j)} \right) \tag{3.5}$$

where $\frac{2}{|V|^2-|V|}$ is the inverse of the cardinality of the possible combinations of pairs within a cluster, while $|X||Y|$ is the value resulting from combining the elements of two different communities.

Embedding base distance. The structural information component is represented by the vector space defined by the map embedding. To compare the vectors, we use the weighted cosine similarity. Below, a mathematical formulation for calculating both *intra-cluster* and *inter-cluster* similarities:

$$E_d = \frac{2}{|V|^2 - |V|} \sum_{\vec{v}_i, \vec{v}_j \varepsilon V} \text{cosine}(\vec{v}_i, \vec{v}_j) \tag{3.6}$$

$$\Delta E_d = \frac{1}{|E||X||Y|} \sum_{i,j}^{c_{\vec{v}_i}=C_X, c_{\vec{v}_j}=C_Y} \text{cosine}(\vec{v}_i, \vec{v}_j) \tag{3.7}$$

where $\vec{v}_i, \vec{v}_j$ are the embeddings of nodes i and j.

In order to obtain the final clustering solution, 4 different steps are performed in sequence:

1. **Hierarchical aggregation.** In the first step, a hierarchical aggregation of markers is performed to capture the clustering structure of the graph and reduce the number of communities. The gain of each community aggregation, which is evaluated according to the gain $\delta\Delta Q + \beta\delta I_d + \gamma\Delta E_d$ (Equation (3.3), (3.5) and (3.7)), is calculated at every iteration.

   Once the values of all possible combinations have been calculated, the two communities with the highest gain are identified. If the gain is positive, the two corresponding communities are merged. The process of calculating the gain and merging the communities is repeated as long as the gain is greater than 0.

2. **Marker relocation.** The second component of the algorithm is responsible for verifying that each community is composed of at least a number of elements equal to a certain threshold. Therefore, if the number of elements assigned to the cluster is lower than the threshold, the markers belonging to it are reassigned to the other existing communities.

   For each cluster element to be reassigned, a new target cluster is identified among the adjacent ones; the choice falls on the one with highest gain $\delta\Delta Q + \beta\delta I_d + \gamma\Delta E_d$ (calculated using the node to relocate as a single-node cluster).

3. **Reduction of the number of clusters.** The third part of the algorithm reduces the number of clusters obtained from the previous steps until the target number of communities is reached. The operations performed are the same as in the hierarchical aggregation step; however, the process halts when the target number of clusters is reached. Notice that in this context, the community recombination always leads to a negative gain; in this case, the cluster merging with the lowest loss is performed.

4. **Handling isolated markers.** The final section of the algorithm deals with the identification of markers that are isolated, i.e. nodes that do not have any neighbors belonging to the same cluster. Once the isolated markers have been identified, two different situations can be profiled: *i)* the marker is surrounded by a single community and is aggregated to that cluster, *ii)* the gain with each of the neighboring clusters is calculated, and the marker is assigned to the community with the highest gain value.

Figure 3.2 shows the four steps that are presented above, where the white circles inserted in Figures 3.2a and 3.2b have been inserted to highlight the change in the communities of these markers with the application of the marker relocation step. In Algorithm 1, the pseudocode of the proposed algorithm is shown in more detail. Moreover, its implementation is available on GitHub [57].

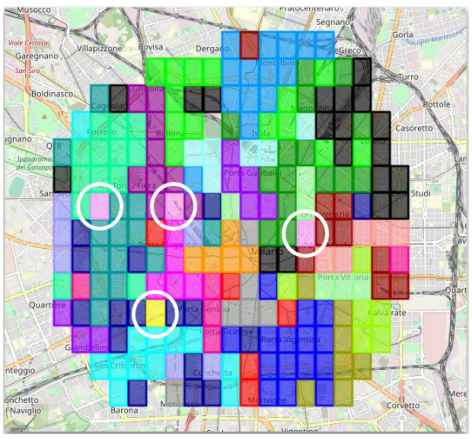
Algorithm 1: Urban Communities Cluster Algorithm

Input: Graph G
Output: C^*
// Hierarchical Aggregation
Initialize each node as an independent community
repeat
  for *each pair* (C_x, C_y) **do**
    $\Delta Q, \Delta I_d, \Delta E_d \leftarrow$ gains(C_x, C_y) // Eq. (3.3), (3.5), (3.7)
  $x_{max}, y_{max} \leftarrow argmax(\delta\Delta Q + \beta\delta I_d + \gamma\Delta E_d)$
  merge communities $C_{x_{max}}, C_{y_{max}}$
until *max* $(\Delta Q + \Delta I_d + \Delta E_d) \leq 0$
// Marker relocation
for *each* $C_y \leq$ *Threshold* **do**
  for *each pair* (C_x, c_{v_i}) *where* $c_{v_i} \in C_y$ **do**
    if $C_x \neq C_y$ **then**
      $\Delta Q, \Delta I_d, \Delta E_d \leftarrow$ gains(C_x, C_y)
  $i_{max}, d_{max} \leftarrow argmax(\delta\Delta Q + \beta\delta I_d + \gamma\Delta E_d)$
  merge communities $c_{v_{i_{max}}}, C_{x_{max}}$
// Reduction of the number of clusters
repeat
  if $|C| >$ *number of clusters* **then**
    for *each pair* (C_x, C_y) **do**
      $\Delta Q, \Delta I_d, \Delta E_d \leftarrow$ gains(C_x, C_y)
    $i_{max}, d_{max} \leftarrow argmax(\delta\Delta Q + \beta\delta I_d + \gamma\Delta E_d)$
    merge communities $C_{x_{max}}, C_{y_{max}}$
until $|C| =$ *number of clusters*
// Relocation of isolated markers
for *each* $c_{v_i} \in C_y$ **do**
  if c_{v_i} *is isolated* **then**
    if c_{v_i} *is surrounded only by* C_x **then**
      merge c_{v_i}, C_x
  else
    for *each* $C_x \in N_{c_{v_i}}$ **do**
      $\Delta Q, \Delta I_d, \Delta E_d \leftarrow$ gains(C_x, c_{v_i})
    $i_{max}, d_{max} \leftarrow argmax(\delta\Delta Q + \beta\delta I_d + \gamma\Delta E_d)$
    merge communities $c_{v_{i_{max}}}, C_{x_{max}}$

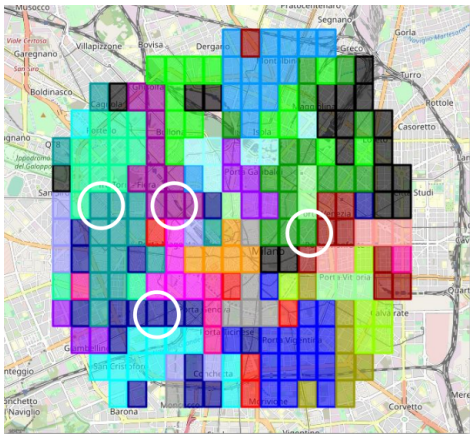
3.3 Related Work

3.3.1 Location Embedding

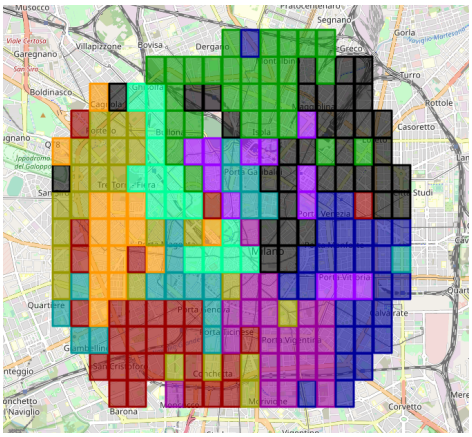
The analysis of environmental information mainly in an urban context is rich in structural, geographical and commercial information can be extracted from a satellite image, defined as *map tile*, the representation of which, however, is generally very computationally intensive. Therefore, the efficient use of this extracted information, through learning the representation for *location embedding* (with size reduction) is currently a



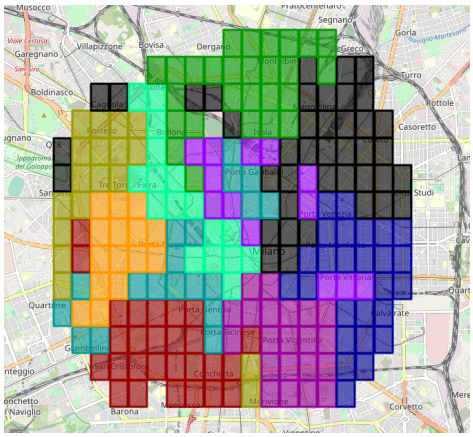
(a) Hierarchical aggregation



(b) Marker relocation step



(c) Cluster reduction step



(d) Isolated point step

Figure 3.2 Example of the application of the four steps for the creation of a cluster in the city of Milan

much studied topic. In [213] a new method is proposed to transform GPS coordinates, which are fine-grained position indicators, into semantic feature vectors through a new structure based on a two-level grid to learn semantic embeddings for geo-coordinates from around the world. In [151], a Siamese type embedding model was trained using map images and Google Street View images. While, [107] used convolutional autoencoder and Principal Component Analysis (PCA) to generate location embedding from Street View images, but these do not contain semantic information. Some state-of-the-art works such as [92, 171] assume that neighboring locations have similar semantic representations, an assumption that may be incorrect since in a dense urban area, geospatial information may vary significantly between them, despite two locations being very close to each other. On the contrary, [202] defines similarity by incorporating POI metadata to the map images. Finally, [200] proposes an approach that uses geographic maps and structural metadata collected by Open Steet Map. Map tiles describe the map in its geospatial information such as shapes, colors and size of structures, while metadata describes it by numerical values, such as the number of streets, parks and buildings. These two different pieces of information are used to construct a supervised multi-label classification problem.

3.3.2 Communities Identification

The aim in identifying the structure of urban communities is to study the internal characteristics and mechanisms of urban communities [196]. Over the years, two categories of research have received particular attention: *functional regions* [209], which is a territorial unit resulting from the organization of social and economic relations, and population flow [145]. In recent years, Point of Interest (POI) have been widely used in the literature as they are easy to access and carry regionally relevant functional information. For this reason, researchers in [221] have classified the functions of urban areas using POIs exclusively. In particular, some studies such as [160] analyze different typical patterns of human behavior in cities to find out whether they can help identify the semantics of geographical locations. In line with this branch of research, the relationships between human mobility and geographical areas are explicitly explored in [86]. The studies analyzed so far in this subsection have verified the feasibility of analyzing the interaction between geographical areas through population flow.

Hierarchical clustering is one of the main methods to identify urban community structure in machine learning, as it allows studying the information of the graph structure [91]. Hierarchical clustering has two advantages over K-means: *i)* it does not need to predetermine the number of clusters as it is not known in advance how many communities can be identified in the analyzed area and *ii)* it has the characteristics of fast speed. In [23], a hierarchical cluster is applied: the community after each iteration is considered as a new node, generating a new higher order graph structure to continue the iteration until the end. In [168] they propose a combo algorithm that combines three different types of node-level adjustment and allows to remedy the defect of incorporating a local optimum. Finally, in [120], they formulate the problem of identifying the urban community structure as an optimization of the modularity of the graph with constraints on the distribution of public services, developing an algorithm using hierarchical clustering combined with node-level adjustment strategies.

Our method is inspired by hierarchical aggregation, but different node-level adjustment techniques are developed, and we introduce an innovative use of map embedding.

3.4 Experimental Analysis

The experimental evaluation has been conducted on a real-world case study. To conduct the experiment, two datasets of the city of Milan are used:

- **Motorbike sharing dataset.** The dataset contains records of vehicle pick-ups and drop-offs from different companies over one week (December 2-8, 2019). The coordinates of each sampled vehicle, the sampling time and the reference marker, i.e. the one closest to the vehicle coordinates, are provided. Sampling has been performed at regular intervals of 15 minutes and the dataset consists of 224 reference markers. By analyzing the dataset, it is possible to trace the trips made by the vehicles and build up a transition matrix that contains travel probability on both weekdays and weekends.

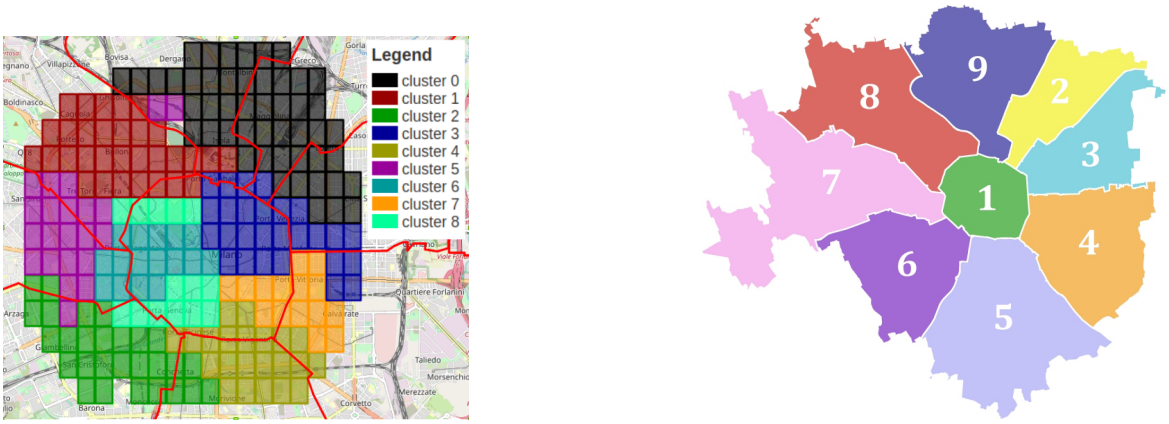
- **Milano Map dataset.** The dataset consists of 224 map tiles, images of size 230×325, and 22 metadata describing some structural information (number of buildings, presence of parks, etc.), for each image. The dataset has been built using entirely the data collected by Open Street Map[1]. The city of Milan has been divided into 9 administrative municipalities in 1999. As shown in Figure 3.3b, the subdivision implies that each municipality, except for the central area, extends from the semi-central zone to the

[1] https://www.openstreetmap.org/#map=13/45.4762/9.1910&layers=C

periphery. The decision to identify the different municipalities following this radio-centric structure reflects the development model of the city that from the single-center projects outward the radials that connect it with the territory.

The peculiarity of our algorithm lies in its ability to obtain dynamic and purpose-specific solutions. In fact, it is possible to control the impact of the three factors (*Modularity*, *Inverse Distance* and *Embedding base distance*) on the creation of the clusters or to choose whether to jointly include trips made during weekdays and weekends or exclude one of the two.

Specifically, by setting the number of clusters to 9 (like the city's municipalities) and assigning more weight to the *Inverse Distance* (component Equation (3.5)), it is possible to obtain a representation of the city that partly follows the subdivision decided by the municipality as shown in Figure 3.3. In fact, municipalities 4 to 8 are quite well identified by our clusters, while municipality 9, 2, and partially 3 are considered to be the same cluster. The municipality 1 represents the center of Milan can be divided in two parts: the first one represented by clusters 1, 4, 6, 7 and 8 that are connected to municipalities 4 to 8, while the cluster 3 covers municipalities 2, 3, and partially 9 (this pattern will be better explain later the discussion).



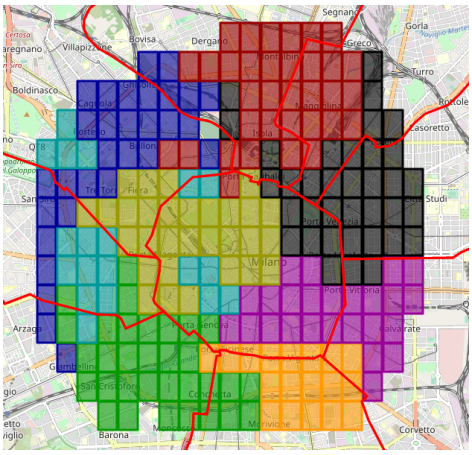
(a) Weighted Inverse Distance

(b) Municipalities of Milan

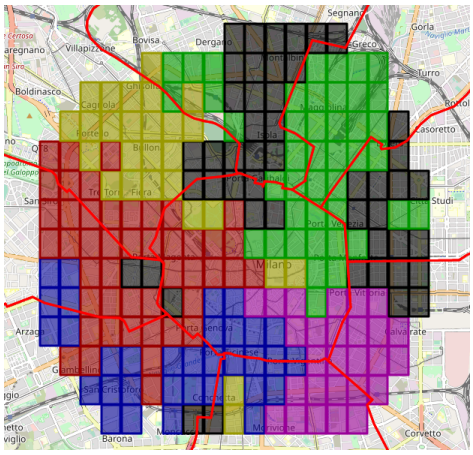
Figure 3.3 Comparison between weighted Inverse Distance and Municipalities

On the contrary, when we give more weight to the *Embedding base distance*, and we analyze the result obtained by considering the weekly and weekend transition matrix separately, it can be seen from Figure 3.4 that the number of clusters identified by the algorithm drops to 8 in the first case and to 6 in the second case. This phenomenon can be explained above all by the diversification of behavior between the two time periods taken into consideration: during the week there is a greater diversification of movements, on the contrary at weekends movements are directed towards the more lively areas of the city. Moreover, in Figure 3.4b, municipality 2 is clearly differentiated from municipality 9 and 3. This is explained due to the number of relevant area for nightlife such as "Garibaldi-Isola" district and shopping area such as "Corso Buenos Aires" that is the largest commercial street in Italy. Such areas are very crowed during the weekend more than weekday. Notice that the observation of this analysis and the previous one, demonstrate that the city division created in 1999 reflects the different area of the city of Milan according to different criteria that can be easily identified by our solution.

Finally, by assigning greater weight to *Modularity* and setting the number of clusters to 9, the communities identified by our algorithm are more fragmented, as shown in Figure 3.5. This effect is the direct consequence of possible differences in travel behavior that can be evidenced even within the same municipalities. By considering the center of Milan, you can see that is can be divided in two halves. The first one represents cluster

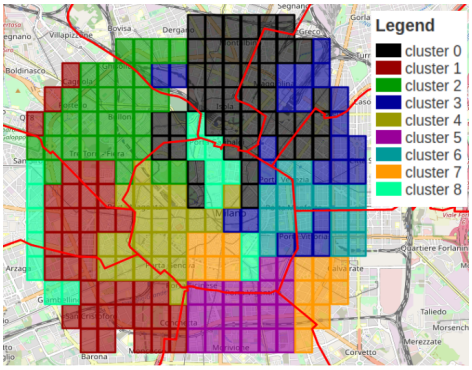


(a) Weekday



(b) Weekend

Figure 3.4 Resulted with weighted Embedding base distance



(a) Weighted Modularity

(b) Municipalities of Milan

Figure 3.5 Comparison between weighted Modularity and Municipalities

2, 4 and 7 that are connected to the municipality 5, 6, 7 and 8; while cluster 0, 3, 6, and 8 intersect the other municipalities.

3.5 Conclusion

Identifying the community structure of a city is a challenging task. In this study, we proposed a clustering algorithm that uses information about mobility (transition matrix), structural information (map embedding) and distance between areas (distance matrix) to identify homogeneous city regions. The developed algorithm has obtained good results in the identification of urban communities, succeeding in obtaining homogeneous areas and identifying consistent and coherent aggregations in the experiment conducted. In fact, the ability of our algorithm to optimize the three different aspects simultaneously allows us to avoid the problems of geographical fragmentation and isolated nodes. Furthermore, the addition of the map embedding component, which integrates the visual and semantic components of the maps in the clustering, allows for a more effective representation of the urban environment, compared to the use of structural metadata alone.

Future Work. It would be interesting to share the results obtained with a domain expert, such as an urban planner, so that he or she could evaluate the solutions obtained. In addition, it would be appropriate to apply the algorithm to other cities and compare it with other clustering techniques such as K-means and Spectral Clustering. For what concern the future development of the algorithm, we should improve the current cluster reduction step by introducing the possibility to increase the number of communities up to the desired value.

4

Displacement Prediction

In recent years, studying and predicting mobility patterns in urban environments has become increasingly important as accurate and timely information on current and future vehicle flows can successfully increase the quality and availability of transportation services (e.g., sharing services). However, predicting the number of incoming and outgoing vehicles for different city areas is challenging due to the nonlinear spatial and temporal dependencies typical of urban mobility patterns. In this chapter, we propose 3D-CLoST and STREED-Net, two Deep Learning models that effectively captures and exploits complex spatial and temporal patterns in mobility data for short-term flow prediction problem. The results of a thorough experimental analysis using real-life data are reported, indicating that the two proposed model bring benefict to the state-of-the-art for this task. Research reported in this chapter has been published in [59, 62].

4.1 Introduction

In recent years, academia and industry have devoted much time and energy to the study and creation of models to describe and predict mobility dynamics, or flow prediction in urban areas. This interest is motivated by the need to comprehend displacement dynamics, which are also rapidly changing due to alternative electric and shared public transport systems, to define effective regulatory strategies for human mobility and freight transport in the smart city [228]. This rush to create increasingly accurate predictive models is also motivated by the pursuit of enhancing the quality of services provided to citizens by both private companies, such as shared mobility companies, and public administrations. As an example, private companies offering shared vehicles can benefit from accurate models to estimate demand in order to improve vehicle relocation operations [178]. On the other hand, the public decision maker can rely on real-time flow data and deep learning models to swiftly identify risky traffic conditions [220].

The problem addressed in this work concerns the short-term flow prediction. More precisely, given a tessellation of the area of interest in squared regions, the number of vehicles entering (Inflow) and exiting (Outflow) each region is to be predicted for the next time period. This problem has inherently spatio-temporal characteristics; evidently, the vehicular flow entering (exiting) a region does not only present temporal dependencies (time of day, flow in the previous hours) but also spatial dependencies as it strongly depends on the traffic leaving (entering) adjacent areas. Formally, such considerations relate to two widely recognized properties in the study of displacement dynamics [223], namely temporal and spatial correlations. Mobility data are innately continuous time series, generally not associated with abrupt changes. This means that the displacement dynamics in periods temporally close share similarities, and this phenomenon is all the more true when the sampling frequency increases. Similarly, since the outflow of an area constitutes the inflow of its neighbors (and vice versa), there is a manifest spatial correlation in traffic dynamics that is widely recognized and exploited in the literature. The most recent research has also shown that some spatial and temporal patterns influence forecasting more than others. This is the case for some *districts* [122], conglomerate areas featuring similar functional characteristics (e.g., residential, commercial and industrial areas), that show correlated traffic patterns and explain much of the city's traffic. Finally, external factors also have a profound impact on the use of vehicles. For instance, it is well known in the literature that weather conditions and the days of the week (workdays vs. weekend) affect displacement dynamics, especially for lightweight transport means like bikes [223].

The rest of this chapter is organized as follows. Section 4.2 defines the flow prediction problem in urban areas. The first proposed Deep Learning model is described in Section 4.3, while the second architecture is presented in Section 4.4. In Section 4.5, the literature on techniques used for flow and traffic prediction is reviewed. In Section 4.6 data and results of experiments are presented and analyzed, and in Section 4.7 a specific case study is introduced. Finally, conclusions and recommendations for future work are discussed in Section 4.8.

4.2 Problem Statement

Given a tessellation of the area of interest (henceforth referred to as *city*) in regularly-shaped regions, a set of historical observations regarding trajectories of vehicles within the city and, possibly, other spatial and non-spatial data sources for a reference time horizon T_H of H time points, the *citywide vehicle flow prediction problem* [34] is defined as the problem of minimizing the prediction error for vehicle *Inflow* and *Outflow* at time t' that is the first time point after T_H.

In the literature, there are several definitions of location/region with different granularity and different semantic meaning [98]. However, when it comes to traffic forecasts, the majority of works use a rectangular tessellation, which maximizes the number of neighboring areas. Similarly, in this study, the geographical space of interest (city) is logically partitioned into a regular grid of size $N \times M$ oriented by longitude and latitude [223]. Each element of the grid is termed *region* and is addressable through a pair of coordinates (n,m) corresponding to the nth row and the mth column of the grid.

The term Inflow (Outflow, respectively) refers to the number of vehicles entering (leaving) a specific region in the considered time unit (Figure 4.1) [34]. More specifically, the Inflow (Outflow) indicates the number of pedestrians, cars, public transport, and sharing vehicles entering (leaving) the region in a certain time period. As shown in Figure 4.1, by analyzing the movement data of the vehicles, it is possible to obtain the Inflow and Outflow matrices, which encompass the information about displacements between the areas of the city at each time t. More in detail, let $\tau_i = \{s_i^1, s_i^2, \dots s_i^t\}$ be a trajectory where s_i^t represents the position of vehicle i at

time t, and let $\mathscr{T}$ be a collection of trajectories. The Inflow (Outflow, respectively) of a region (n,m) at time t, namely $\iota^t_{n,m}$ ($\omega^t_{n,m}$) can be formally defined as in Equations (4.1) and (4.2), respectively).

$$\iota^t_{n,m} = \sum_{\tau_i \in \mathscr{T}} \phi_\iota(\tau_i, t, n, m) \tag{4.1}$$

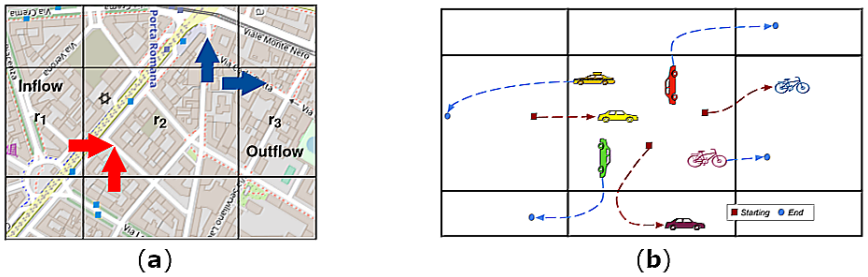
$$\omega^t_{n,m} = \sum_{\tau_i \in \mathscr{T}} \phi_\omega(\tau_i, t, n, m) \tag{4.2}$$

where

$$\phi_\iota(\tau_i, t, n, m) = \begin{cases} 1, & \text{if } s_i^{t-1} \notin (n,m) \wedge s_i^t \in (n,m) \\ 0, & \text{otherwise.} \end{cases}$$

and

$$\phi_\omega(\tau_i, t, n, m) = \begin{cases} 1, & \text{if } s_i^{t-1} \in (n,m) \wedge s_i^t \notin (n,m) \\ 0, & \text{otherwise.} \end{cases}$$



(**a**) (**b**)

Figure 4.1 Measurement of flows. (**a**) Inflow and Outflow; (**b**) Measurement of flows.

Finally, the state of the vehicular flow at time t can be represented by a tensor (also referred to as *frame* in what follows) $F_t \in \mathbb{R}^{N \times M \times C}$, where C indicates the number of flow variables considered in the analysis, in this specific case $C = 2$ (Inflow/Outflow), whereas $N \times M$ is the total number of regions in the city. Then, to take into account the temporal dependence, over the time horizon T (divided into H time points), the flow representation is extended to a tensor of four dimensions $F \in \mathbb{R}^{H \times N \times M \times C}$, which represents the main input to our problem. The problem at issue then becomes predicting F_t given a *volume*, that is a sequence of past tensors $\mathscr{V} \subset F$. It is worth noting that the resulting problem shows several similarities with the frame prediction problem [59] since the tensor F can be seen as a four-dimensional volume composed of H consecutive images, each of which featuring C channels.

4.3 3D-CLoST

The first proposed vehicle flow prediction framework is 3D-CLoST (3D **C**onvolution **L**STM **o**n **S**patio - **T**emporal). It exploits the synergy between 3D convolution and long short-term memory (LSTM) networks to jointly learn the characteristics of the space-time correlation in urban mobility. The main innovations and contributions of 3D-CLoST are as follows:

- To the best of our knowledge, 3D-CLoST is the first attempt to combine 3D convolutional and LSTM nets to predict mobility dynamics in urban environments.

- The use of autocorrelation analyses is proposed for cherry-picking a reduced set of historical data with multi-temporal correlation. It allows us to build a framework that does not require the use of multiple branches to model the different time dependencies.
- The use of a mask is proposed to coerce the model to properly account for areas that are forbidden to vehicles (for instance river areas).

4.3.1 3D-CLoST Architecture

In this subsection, the proposed vehicle flow prediction framework (3D-CLoST) is presented and discussed. Figure 4.2 displays the two main blocks of 3D-CLoST architecture, that are:

- **Core component:** A Deep-learning model featuring a composition of different types of neural networks. In a nutshell, a 3D CNN is used to learn spatial and temporal patterns, followed by a LSTM to strengthen the temporal aspect. A fully connected network, which may considers also external inputs, completes the model (see Figure 4.3.1 for more details).
- **Extension:** A set of corollary techniques (discussed in Equation 4.3.1) developed to streamline the construction of small-sized input volumes (a sample of D along the time axis containing only the most significant data points) and to enforce the framework to handle prohibited regions.

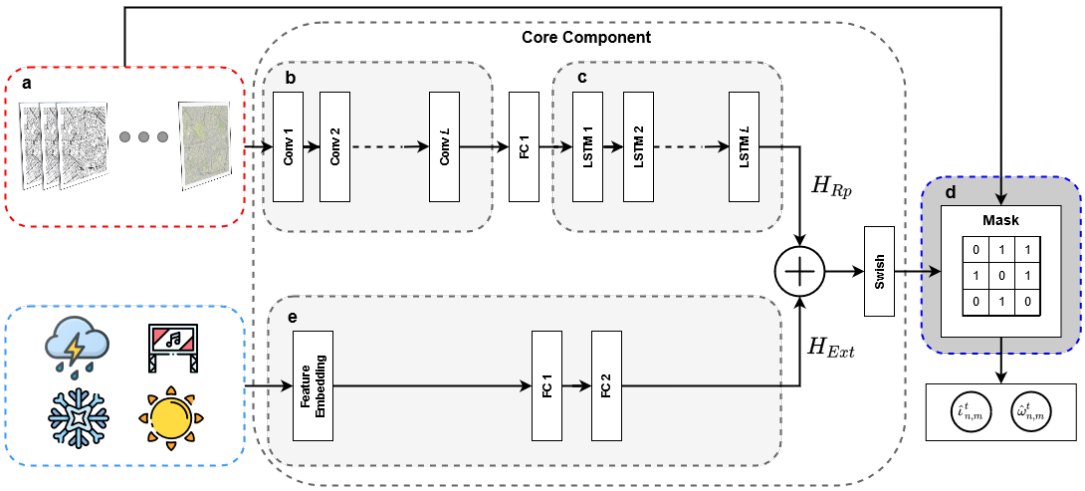


Figure 4.2 The architecture of 3D-CLoST. (a) Heuristic for the creation of the volumes. (b) Spatial and temporal dependencies are captured by the convolutional stage, which subsequently pass into the (c) LSTM section. (e) Transform external information and insert it into the model. (d) The mask applied to the output of the neural network

Core Component

As mentioned, the core of 3D-CLoST, available under an open source license on GitHub [58], consists of a convolutional network, followed by an LSTM, and concluded by a fully connected network that may consider external factors to generate inflow and outflow forecasts. The number of convolutional and LSTM layers is dictated by factors such as the city size and vehicle type.

Our framework implements a 3D convolutional layer, which captures temporal dependencies better than a 2D one [183] that, if applied on $\mathscr{V}$, would compress the temporal axis immediately after each convolution operation, causing a critical information loss.

More in detail, the 3D CNN consists of multiple stages intended to extract information from the tensor with increasing levels of abstraction. Each of them is made of three cascading layers: *convolutional layer*, *activation layer*, and *pooling function*. The three stages are described below.

Convolution layer: this stage performs a 3D convolution (on both flow variables) between the input tensor and a kernel:

$$S(n,m,t) = (\mathscr{V} * K)(n,m,t) \tag{4.3}$$

where $\mathscr{V}$ is the input tensor carrying past flow information, K represents the *kernel*, while the output S is referred to as the *feature map* or kernel map.

Activation layer: this second stage uses a nonlinear activation function to learn complex dependencies. In accordance with the best practices of literature [199], ReLU has been selected as activation function, which considerably accelerates the convergence of the stochastic gradient descent.

Pooling layer: it reduces the spatial size of the feature map extracting dominant features by means of a summarization process performed on the feature map by region. It has the advantage to reduce the computational complexity of the network and to make the network more robust to small variations in the input [29].

In cascade to the convolutional levels a dense, fully connected layer of 128 units with ReLU activation function has been placed, its purpose is to convert and resize the feature map (in input) into a vector of information that can be passed to the next stage, the LSTM one.

The long short-term memory (LSTM) [78] is another first class deep learning architecture used to identify patterns in sequences of data points, it has been introduced in the framework to improve the ability of capturing temporal dependencies. More in details, it is a recurrent neural network with a sequence of interconnected units, referred to as *cells*. Each cell acts as a memory that keeps track of the dependencies between the points of the input. Within each cell, three main structures, called gates, can be found: an *input gate*, a *forget gate*, and an *output gate*. Those three gates, which have independent weights and biases, are meant to drive the network to learn:

- How much of the current input to use to update the cell state (input gate)
- How much of the cell state to be forgotten (forget gate)
- How much of the cell state to use to generate the output activation (output gate)

Stacked LSTM architecture has been implemented in 3D-CLoST. It can be defined as an LSTM model comprised of multiple LSTM layers. The use of multiple layers improves the effectiveness of the network in recognizing complex patterns, potentially allowing the hidden state of each layer to operate on a different timescale. Technically, a stacked LSTM is implemented by linking various LSTM layers, each containing several cells. Such a network outputs a tensor (with three dimensions, where the first dimension corresponds to the batch size) rather than a single matrix.

Further, 3D-CLoST encompasses a stage dedicated to external factors; those can deeply affect the vehicular flow. Examples of such factors are weather conditions (rainfall and temperature), traffic events, and the day of the week (workdays, weekend). Through the use of two stacked fully connected layers, this information is conveyed, encoded, into the main flow of the network. The first layer serves to embed each sub-factor whereas

the second one is used to map low to high dimensions in order to match the size of the flattened LSTM output vector, a similar approach has been used in [34]. Further downstream, in fact, the LSTM output vector (H_{Rp}) and the external components (H_{Ext}) join:

$$H_f = H_{Rp} \oplus H_{Ext} \tag{4.4}$$

Finally, the joined output H_f becomes the input of a fully connected layer using the Swish activation function, defined:

$$F(x) = x \times \sigma(x) \tag{4.5}$$

where $\sigma(\cdot)$ is the sigmoid function.

We conclude this section with a brief analysis of the complexity of the model. To achieve the general complexity, it is necessary to analyze the convolutional layers and the LSTM layers individually [185]. The complexity of all the convolutional layers can be estimated as $O(\sum_{l=1}^{d} n_{l-1} \cdot s_l^2 \cdot n_l \cdot m_l^2)$, where d is the number of convolutional layers, n_l is the number of filters in the l^{tj} layer, n_{l-1} is the number of input channels of the l^{tj} layer, s_l is the spatial dimension of the filter and m_l is the spatial dimension of the map of the output characteristics. Regarding the LSTM layers, the time complexity per weight is $O(1)$, with an overall per time step equal to $O(w)$. Consequently, the 3D-CLoST complexity for all the training process is $O((\sum_{l=1}^{d}(n_{l-1} \cdot s_l^2 \cdot n_l \cdot m_l^2) + w) \cdot i \cdot e)$ where i is the input length and e the number of epochs.

Extension

As introduced previously, the problem addressed in the study concerns the prediction of F_t using a volume $\mathscr{V} \subset F$. Creating a minimum size volume that includes only the frames prior to time t, that express most reliably the time dependencies in the displacement dynamics, and guarantees both low prediction error and reduced training times is undoubtedly an appealing topic. In this section, we identify the various types of time dependencies and propose a heuristic for creating $\mathscr{V}$ from historical data.

Within the historical series, it is possible to identify two types of temporal dependencies, which can be roughly classified as near and far. The vehicular flow dependence on nearby periods is responsible for the flow trend, while a possible relationship with more remote periods describes recurrent phenomena. In urban areas, the main seasonalities are daily and weekly; thus, distant time dependencies can be further specified in two sets of frames referring to two periods called Recent and Distant, respectively. In what follows, those three periods are precisely defined (see also Figure 4.3).

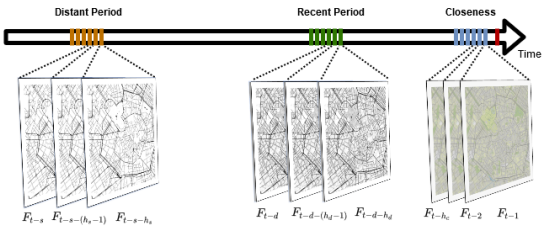


Figure 4.3 Overview of the frame selection

Nearby periods (also called *closeness* [34]) are selected by setting a maximum number of time slots h_c before the time to predict t, it results in a sequence $[F_{t-h_c}, F_{t-(h_c-1)}, ..., F_{t-1}]$ of consecutive frames.
For the far periods, we have:

Algorithm 2: Volume Construction Heuristic

Input : F; // inflow-outflow information
thr; // threshold
Output : $\mathscr{V}$; // volume

$\rho \leftarrow autocorrelation(F)$
recent_period $\leftarrow$ checkRecentPeriod(ρ)
distant_period $\leftarrow$ checkDistantPeriod(ρ)
$(h_c, d, h_d, s, h_s) \leftarrow$ identifyRelatedHours(ρ, thr)
for *i in [1, h_c]* **do**
  $\mathscr{V}_i \leftarrow F_{t-i}$
if *(recent_period == True)* **then**
  for *i in [0, h_d]* **do**
    $\mathscr{V}_i \leftarrow F_{t-d-i}$
if *(distant_period == True)* **then**
  for *h_s in [0, h_s]* **do**
    $\mathscr{V}_i \leftarrow F_{t-s-i}$
return $\mathscr{V}$

- *Recent period.* Let d be the period span and h_d the maximum number of time slots, the recent period is made of frames $[F_{t-d-h_d}, F_{t-d-(h_d-1)}, ..., F_{t-d}]$.
- *Distant period.* Let s be the period span and h_s maximum number of time slots, the distant period is made of frames $[F_{t-s-h_s}, F_{t-s-(h_s-1)}, ..., F_{t-s}]$.

The problem of identifying time dependencies is known in the literature, where it is often tackled via automatic extraction that is the identification of frame relevance is left to the network, relying on the network ability to learn the functionality directly from the raw data. This type of approach imposes the use of a more complex model architecture [34] with different branches based on time dependence (hourly, daily and weekly) and a large number of input frames. As a result, the algorithm turns out to be slower to converge and learn.

Instead of leaving every choice to the network, a feature engineering approach to select the most significant frames is proposed. To thin aim, we have developed a heuristic (algorithm 2) based on the use of the autocorrelation analysis. Autocorrelation is the correlation of a signal with a delayed copy of itself as a function of the considered delay. It is a time-domain measure of a stochastic process memory. More in details, given a pool of measurements e_1, e_2, e_2, ..., e_n at time t_1, t_2, ..., t_n the lag k autocorrelation function $\rho(k)$, for a non-stationary stochastic process, is defined as:

$$\rho(k) = \frac{Cov(e_n, e_{n+k})}{\sqrt{Var(e_n)Var(e_{n+k})}} \tag{4.6}$$

The heuristic (presented in Algorithm 2) develops by setting a threshold (0.3 in the studies we have carried out) above which the frames are considered carrying important information to predict flows at time t. The size of the initial volume will depend on the chosen threshold value. Nonetheless, the volume structure can still be improved, for this reason, we have explored alternative configurations around this initial solution (see Appendix A.1 for more details). Nevertheless, the following reasons lead to an exploration of the neighborhoods in the initial volume:

i. The awareness that there may be differences among the city regions

ii. The autocorrelation is a linear analysis; thus, it might neglect possibly important non-linear correlations

This approach has allowed us to build a single tensor to model the different temporal relationships (near and far), significantly reducing the number of parameters to be learned.

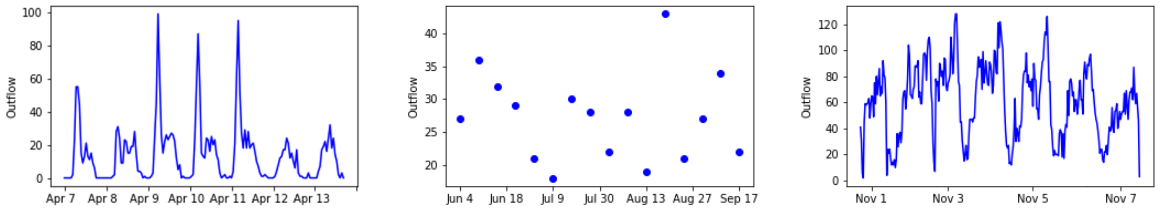
After discussing time constraints, we consider another important aspect of the flow prediction problem, which concerns traffic forbidden regions i.e. those regions that due to the conformation of the territory (lakes, rivers, mountains, etc.) and urban planning, cannot be associated with certain type of vehicle. For example, considering New York City, the data shows that bikes can never be found in 62.5% of the town areas, so the number of bicycles in those areas will always be zero. The main effect of this phenomenon concerns the problem of the class imbalance [90]. In fact, when the number of regions, in which there are never bikes, is high, there is a greater difficulty for the neural network in better predicting those areas that have a number of bicycles that varies between [0, N] [30]. However, there is no indication in the literature on how to manage such regions; thus, a simple but effective solution has been implemented to avoid that the regions without vehicles affect the results of the model. It has been decided to insert a mask of values $\{0; 1\}$ calculated over the historical data. The mask is a tool to break the continuity of each frame, forcing the convolutional kernels to drop specific frame cells. The output of the core component is multiplied to as follows:

$$\hat{o}^t_{n,m} = \begin{cases} \hat{y}^t_{n,m} \cdot 0 & \text{region where there can never be vehicles} \\ \hat{y}^t_{n,m} \cdot 1 & \text{otherwise} \end{cases}$$

where $\hat{o}^t_{n,m}$ is the vehicular flow predicted for region (n,m) at time t, $\hat{y}^t_{n,m}$ is the output of the fully connected network. The mask is created in two steps:

1. Computation of the magnitude (number of vehicles) for each region of the city for each period t.
2. Sum of all magnitudes over the whole reference time horizon.

If the sum of a region is equal to zero, then the value of the mask is set to zero. Notice that, the mask is an integral part of the neural network and performs its function during training. The mask, in its simplicity, has proved effective, reducing the Root Mean Square Error (RMSE) of approximately 1.28% in the experiments carried out.



(a) (b) (c)

Figure 4.4 (a) Bike week; (b) Bike weekly seasonality; (c) Taxi daily seasonality

4.4 STREED-Net

The second proposed vehicle flow prediction framework is STREED-Net (**S**patio **T**emporal **RE**sidual **E**ncoder-**D**ecoder **Net**work). The main contributions of this model are the following:

- To the best of our knowledge, STREED-Net is the first autoencoder architecture that combines the use of time-distributed convolutional blocks with residual connections, a CMUs and two different attention mechanisms.
- STREED-Net, unlike other state-of-the-art models [34, 193, 223], focuses only on recent time dependencies (closeness), by taking into account a few previous time periods. This simplifies the prediction process by requiring less information to be considered.

Figure 4.5 shows the architecture of the STREED-Net: it is an Autoencoder Deep Learning model that combines time-distributed convolutions and CMU with two different types of *Attentions* (spatial and temporal). Therefore, after a short introduction to the main underpinning concepts, namely the autoencoder architecture and the attention mechanism, the components of STREED-Net (Encoder, External Factors and Decoder) are presented in details.

4.4.1 Background

An overview of autoencoder architecture and the attention mechanism is presented in this subsection.

Autoencoder architecture. Given a set of unlabeled training examples $\{x^1, x^2, x^3, ...\}$, where $x^i \in \mathbb{R}^n$, an autoencoder neural network is an unsupervised learning algorithm that applies backpropagation setting the target values to be equal to the inputs $y(i) = x(i)$. It is a neural network that is trained to learn a function $h_{W,b}(x) = \hat{x} \approx x$, where W and b are weights and biases of the ANN, respectively. In other words, an autoencoder approximates the identity function by producing $\hat{x}$ that is as similar to x as possible. The overall network can be decomposed into two parts: an encoder function $h = f(x)$, which maps the input vector space onto an internal representation, and a decoder that transforms it back, that is $\hat{x} = g(h)$. This type of architecture has been applied successfully to different difficult tasks, including traffic prediction [212].

Attention mechanism. In Deep Neural Network (DNN) Attention Mechanism helps focus on important features of the input, shadowing the others. This paradigm is inspired by the human neurovisual system, which quickly scans images and identifies sub-areas of interest, optimizing the usage of the limited attention resources [187]. Similarly, the attention mechanism in DNN determines and stresses on the most informative features in the input data that are likely to be most valuable to the current activity. Recently, attention has been widely applied to different areas of deep learning, such as natural language processing [11], image recognition [226], image captioning [210], image generation [70] and traffic prediction [121].

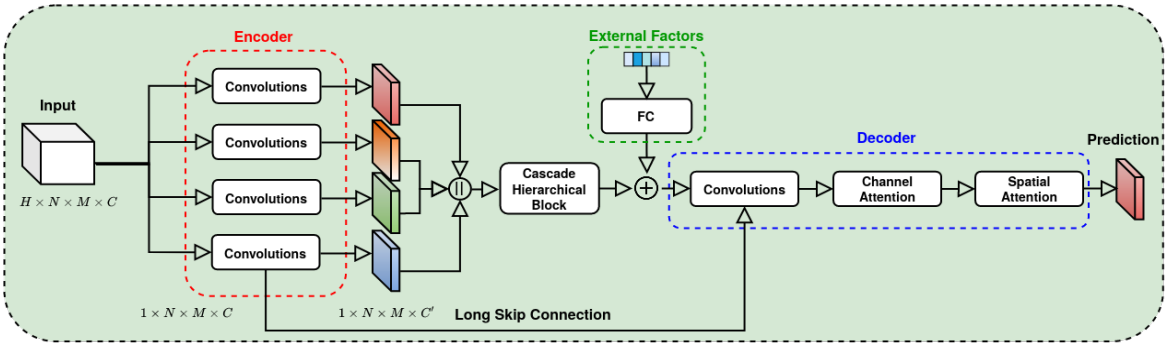


Figure 4.5 STREED-Net Architecture.

4.4.2 Encoder

The encoder structure depicted in Figure 4.6 is the first block of the STREED-Net architecture.

It is composed of an initial convolutional layer, a series of *residual units*, and a final convolution layer. Unlike similar approaches (e.g., STAR [193]), the proposed encoder structure introduces three novel aspects: (i) each layer is time-distributed, meaning that the model learns from a sequence of frames (for time coherence) instead of focusing on each frame singularly. (ii) it applies further convolutions after the residual unit, so that to reduce the frame size and (iii) it applies a Batch Normalization (BN) after each convolution to avoid gradient disappear/explode problems and achieve faster and more efficient reported optimization [85, 154].

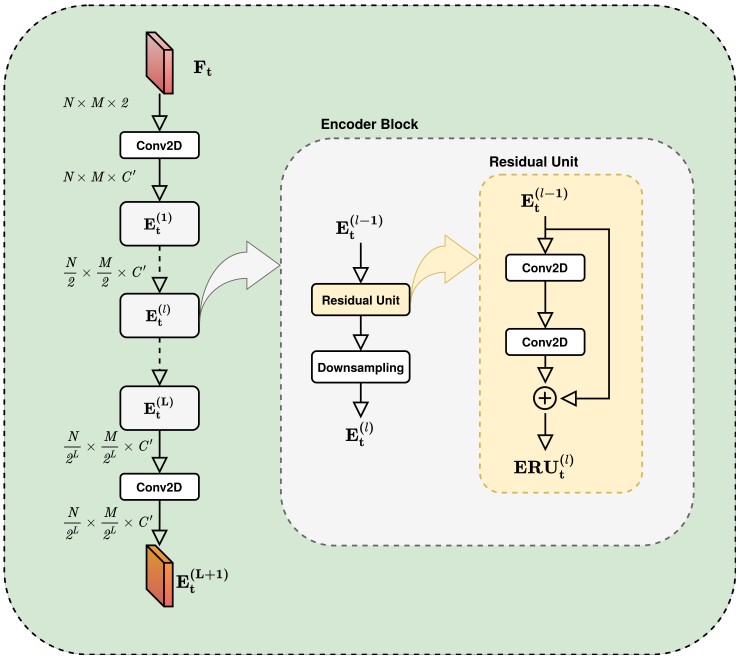


Figure 4.6 Encoder.

Unlike other works from the literature, where the distant temporal information is also used (from the previous day and previous week), the encoder takes as input a four-dimensional tensor $F \in \mathbb{R}^{H \times N \times M \times 2}$. This tensor is a sequence of consecutive three-dimensional frames conveying flow information of nearby periods (with regard to the prediction time t'). Such a tensor (also referred to as *closeness* in the literature [34]) is obtained by selecting p points preceding the prediction time t', i.e., the sequence $[F_{t'-p}, F_{t'-(p+1)}, ..., F_{t'-1}]$. In this way, STREED-Net can focus on the most recent dynamics only. Each frame in F is processed by the convolution layer to extrapolate spatial information. It is worth noting that in Figure 4.5, the encoder is represented by a collection of identical blocks in parallel execution on the input frames instead of (as in reality) a single convolution applied sequentially. Such a representation is used to highlight that a time-distributed layer is trained by taking into account all input frames simultaneously. The use of this approach leads the model to identify temporal (that is, inter-frame) dynamics, rather than looking only to spatial dependencies within each frame.

Each convolutional layer is followed by a ReLU activation function and a *BN* layer. Formally, we have:

$$E_t^{(0)} = BN(ReLU(W_e^{(0)} * F_t + b_e^{(0)})) \tag{4.7}$$

where $E_t^{(0)}$ corresponds to the output of the first convolutional layer, F_t is one of p frames in input to the model and $*$ the convolution operator. $W_e^{(0)}$ and $b_e^{(0)}$ are the weights and biases of the respective convolutional operation. Next, $\mathbf{L}$ encoder blocks (see Figure 4.6) are placed. Each of these blocks is composed of a residual unit followed by a downsampling layer:

$$E_t^{(l)} = Downsampling(ResUnit(E_t^{(l-1)})) \tag{4.8}$$

where $E_t^{(l-1)}$ and $E_t^{(l)}$ correspond respectively to the input and output of the encoder block; l takes values in $\{1,...,\mathbf{L}\}$. The residual units have been implemented as a sequence of two convolutional layers whose output is eventually summed to the block input. Mathematically, in STREED-Net the residual unit are defined as follows:

$$c_1 = BN(ReLU(W_1^{(l)} * E_t^{(l-1)} + b_1^{(l)})) \tag{4.9}$$

$$c_2 = BN(ReLU(W_2^{(l)} * c_1 + b_2^{(l)})) \tag{4.10}$$

$$ERU_t^{(l)} = E_t^{(l-1)} + c_2 \tag{4.11}$$

where $E_t^{(l-1)}$ is the residual unit input and $ERU_t^{(l)}$ (*Encoder Residual Unit*) is used to indicate the result of $ResUnit(E_t^{(l-1)})$. $*$ is the convolution operator, $W_1^{(l)}$, $W_2^{(l)}$ and $b_1^{(l)}$, $b_2^{(l)}$ are the weights and biases of the respective convolutional operations.

For what concerns the *Downsampling*, it has been implemented as:

$$E_t^{(l)} = BN(ReLU(W_{ds}^{(l)} * ERU_t^{(l)} + b_{ds}^{(l)}) \tag{4.12}$$

where $W_{ds}^{(l)}$, $b_{ds}^{(l)}$ and $*$ indicate a convolutional layer with *kernel size* and *stride* parameters set to halve the height and width of the input frame.

The rationale behind the design of this architecture is threefold: (*i*) a deep structure is needed for the model to grasp dependencies not only among neighboring regions but also among distant areas; (*ii*) *Deep* networks are difficult to train as they present both the problem of the explosion or disappearance of the gradient and a greater tendency to overfitting due to the large number of parameters. To try to avoid these obstacles and to make the training model more efficient, we introduced residual units. Finally, (*iii*) the downsampling layers were introduced to ensure translational equivariance [67].

Finally, the encoder structure ends with a closing convolution-ReLU-BN sequence, which has as its main objective to reduce the number of *feature maps*. In this way, the next architectural component (i.e., the Cascading Hierarchical Block) will receive and process a smaller input, reducing the computational cost of the CMU array. The encoder output is:

$$E_t^{(L+1)} = BN(ReLU(W_e^{(L)} * E_t^{(L)} + b_e^{(L)})) \tag{4.13}$$

The output of the encoder is a tensor $E^{(L+1)} \in \mathbb{R}^{H \times N/2^{\mathbf{L}} \times M/2^{\mathbf{L}} \times C'}$, where C' is the number of *feature maps* generated by the last convolution of the encoder.

4.4.3 Cascading Hierarchical Block

A connection section between the encoder and the decoder is provided to handle the temporal relationships among the frames. Unlike what is proposed in other works that combine the use of CNN with the use of RNN such as LSTM [148], STREED-Net implements a Cascading Hierarchical Block with CMU (CMU) [212], which computes the hidden representation of the current state directly using the input frames of both previous

and current time steps, rather than what happens in recurrent networks that model the temporal dependency by a transition from the previous state to the current state. This solution is designed to explicitly model the dependency between different time points by conditioning the current state on the previous state, improving the model accuracy; incidentally, it also reduces training times.

The fundamental constituent of CMU architecture is the MU [96], which is a non-recurrent convolutional structure whose neuron connectivity, except for the lack of residual connections, is quite similar to that of LSTM [78]; the output, however, only depends on the single input frame h. Formally, MU is defined by the following equation set:

$$g_1 = \sigma(W_1 * h + b_1) \tag{4.14}$$

$$g_2 = \sigma(W_2 * h + b_2) \tag{4.15}$$

$$g_3 = \sigma(W_3 * h + b_3) \tag{4.16}$$

$$u = tanh(W_4 * h + b_4) \tag{4.17}$$

$$MU(h;W) = g_1 \odot tanh(g_2 \odot h + g_3 \odot u) \tag{4.18}$$

where σ is the sigmoid activation function, $*$ the convolution operator and $\odot$ the element-wise multiplication operator. $W_1 \sim W_4$ and $b_1 \sim b_4$ are the weights and biases of the respective convolutional gates and W denotes all MU parameters.

CMU incorporates three MU. Unlike MU, CMU accepts two consecutive frames as input to model explicitly the temporal dependencies between them. The more recent frame in time is inputted to a MU to capture the spatial information of the current representation. The older frame is instead processed by two MU in sequence to overcome the time gap. The partial outputs are then added together and finally, thanks to two gated structures containing convolutions along with non-linear activation functions, the output of the CMU (X_t^{l+1}) is generated. CMU is described by the following equations:

$$h_1 = MU(MU(E^l_{(t-1)};W_1);W_1) \tag{4.19}$$

$$h_2 = MU(E^l_t;W_2) \tag{4.20}$$

$$h = h_1 + h_2 \tag{4.21}$$

$$o = \sigma(W_o * h + b_o) \tag{4.22}$$

$$X_t^{l+1} = o \odot tanh(W_h * h + b_h) \tag{4.23}$$

where W_1 and W_2 are the parameters of the MU in the left branch and of the MU in the right branch respectively, W_o, W_h, b_o and b_h are the weights and biases of the corresponding convolutional gates. The cascading hierarchical block uses CMUs to process all frames at the same time (see Figure 4.7):

$$X_{cmu} = CascadeCMU(E^{(L+1)}) \tag{4.24}$$

where $X_{cmu} \in \mathbb{R}^{N \times M \times C'}$.

4.4.4 External Factors

As mentioned at the beginning of Section 4, displacement dynamics are influenced by many complex external factors, such as the day of the week, holidays, and weather conditions. For this reason, following similar

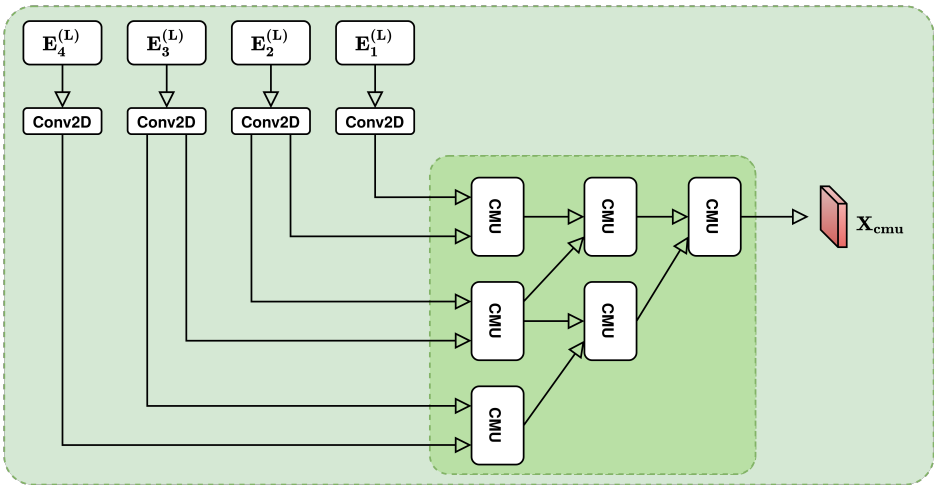


Figure 4.7 Cascading Hierarchical Block.

approaches from the literature [193, 223], features a specific input branch to integrate external information. The input is a one-dimensional vector that contains information that refers to prediction time t'.

Through the use of two fully connected overlapping layers, this information is conveyed, encoded, into the mainstream of the network. The first level is used to embed each sub-factor, while the second reshapes the external factors embedding space to match the size of the *CMU output vector*.

4.4.5 Decoder

The decoder is the last component of STREED-Net and its task is to generate the flow prediction starting from the latent representation that corresponds to the output of the cascading hierarchical block.

As shown in Figure 4.8, the decoder takes as input a tensor $z = X_{cmu} + X_{ext}$, where $z \in \mathbb{R}^{N \times M \times C'}$, which is the result of the sum of the outputs of the hierarchical structure and the network dedicated to incorporate external factors. X_{ext} is added at this point of the network to allow the model to use the information extracted from the external factors during the reconstruction phase.

The decoder architecture features a structure that is somehow symmetrical to that of the encoder with an array of residual units preceded and followed by a convolutional layer (Equation (4.25)).

$$D^{(0)} = BN(ReLU(W_d^{(0)} * z + b_d^{(0)})) \tag{4.25}$$

Nevertheless, this symmetry is breached by the presence of two significant differences. The first one is the presence of a long skip connection before every residual unit. The long skip connection is used to improve the accuracy and to recover the fine-grained details from the encoder. Another benefit is a significant speed-up in model convergence [173].

Generic decoding block $D^{(l)}$, $\forall l \in \{1 \dots L\}$ can be formally defined as the sequential application of the following three operations:

$$sc^{(l)} = Conv2DTranspose(D^{(l-1)}) + ERU_1^{(L+1-l)} \tag{4.26}$$

$$U^{(l)} = BN(ReLU(sc^{(l)}) \tag{4.27}$$

$$D^{(l)} = ResUnit(U^{(l)}) \tag{4.28}$$

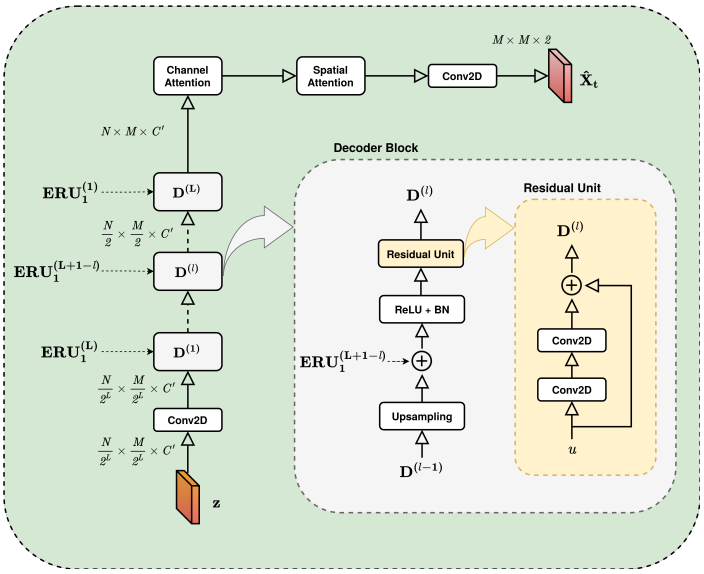


Figure 4.8 Decoder.

where $D^{(l-1)}$ corresponds to the input block, *Conv2DTranspose* indicates the transposed convolution operation (also known as deconvolution), which doubles the height and width of the input, and $sc^{(l)}$ (skip connection) is the sum of u with $ERU_1^{(L+1-l)}$, i.e., the output of the remaining encoder unit at level $L+1-l$ for the most recent frame. The residual units of the decoder are structured exactly like those of the encoder.

The second difference is the presence of two attention blocks (viz. Channel and Temporal Attention) before the final convolution layer. More details are provided in the following subsections.

Channel Attention

After the convolutional stage of the decoder, a three-dimensional tensor, referred to as $D^{(L)} \in \mathbb{R}^{N \times M \times C'}$, is obtained with the channel size C'. Since the dimension of the channel also includes the temporal aspects compressed by the cascading hierarchical block, the channel attention [204] has been introduced to identify and emphasize the most valuable channels. Figure 4.9 depicts the inner structure of the *Channel Attention Block*. Given the input tensor, $D^{(L)}$ a channel attention map $A_c \in \mathbb{R}^{1 \times 1 \times C'}$ is created by applying attention block deduction operations on the channels. More precisely, through the operation of global average pooling and global max pooling performed simultaneously, two different feature maps (X^{max} and X^{avg}) of size $1 \times 1 \times C'$ each are spawned. The rationale behind the choice to use both pooling strategies is that the avg pooling (X^{avg}) allows for the computation of spatial statistics [80], whereas max pooling (X^{max}) provides basic translation invariance to the internal representation by observing the maximum presence of different features.

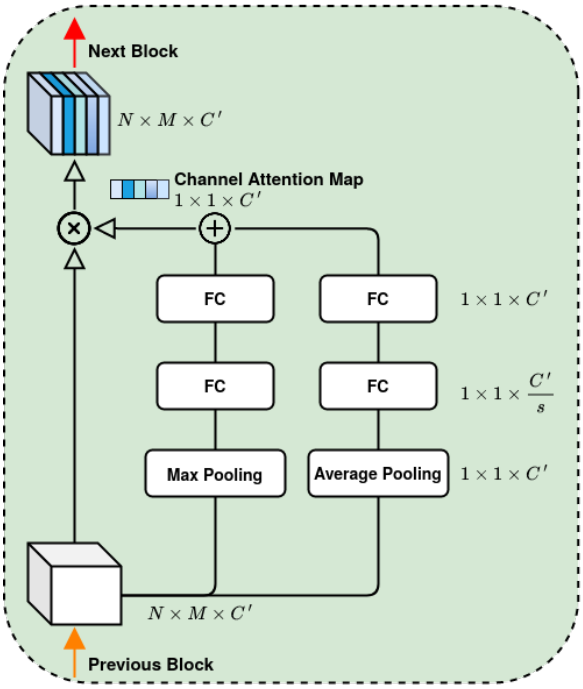


Figure 4.9 Channel Attention Block.

The two feature maps (X^{max} and X^{avg}) go through two Fully Connected (FC) layers that allow the model to learn (and assess) the importance of each channel. The first layer performs a dimensionality reduction, downsizing the input feature maps to $1 \times 1 \times \frac{C\prime}{s}$, based on the choice of the reduction ratio s; the second layer restores the feature maps to their original size. This approach has proven to increase the model efficiency without accuracy reduction [121]. Once these two steps have been completed, the two resulting feature maps are combined into a single tensor through a weighted summation as:

$$\begin{aligned} A_c &= \sigma(\Lambda_1 \otimes FC(FC(X^{max})) + \Gamma_1 \otimes FC(FC(X^{avg}))) \\ &= \sigma(\Lambda_1 \otimes W_2(W_0(X^{max})) + \Gamma_1 \otimes W_3(W_1(X^{avg}))) \end{aligned} \tag{4.29}$$

where σ denotes the sigmoid function, $W_0 \in R^{C' \times \frac{C'}{s}}$, $W_1 \in R^{C' \times \frac{C'}{s}}$, $W_2 \in R^{\frac{C'}{s} \times C'}$ and $W_3 \in R^{\frac{C'}{s} \times C'}$ represent the weights of the FC layers, and Λ_1 and Γ_1 are two trainable tensors with the same size as the two feature maps. Λ and γ are set during the training phase and weight the relative importance of each element of the two feature maps. Finally, the process of getting channel attention can be summarized as:

$$D^{'} = A_c \otimes D^{(L)} \tag{4.30}$$

where $D^{'}$ is the operation output and $\otimes$ denotes the element-wise multiplication.

Unlike its original version [204], the weights of Fully Connected layers are independent of each other, and we add two variables Λ and Γ to enable the network to learn how to best balance the impact of the two branches.

Spatial Attention

Cities are made up of a multitude of different functional areas. Areas have different vehicle concentrations and mobility patterns; thus, the spatial attention mechanism has the task of identifying where are located the most significant areas and scale their contribution to improve the prediction. Figure 4.10 presents the main internals involved in the calculation of the spatial attention map.

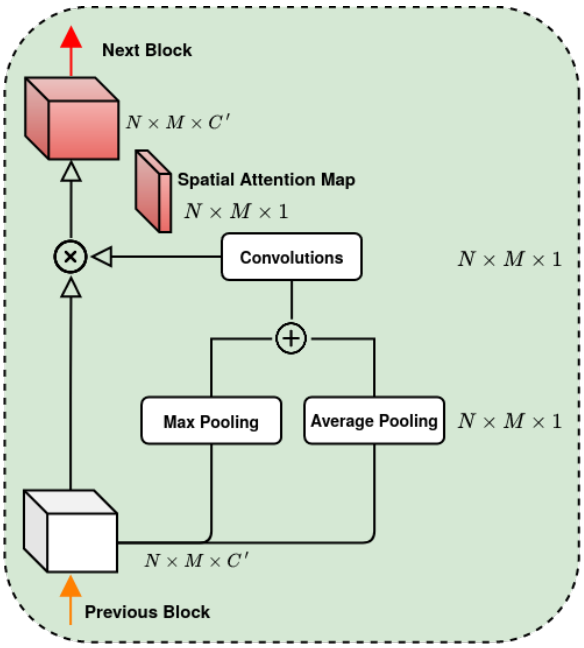


Figure 4.10 Spatial Attention Block

The spatial attention map $A_s \in \mathbb{R}^{N \times M \times 1}$ can be calculated by applying pooling operations along the axes of the channel to highlight informative regions [102]. Therefore, first the global average pooling (X^{avg}) and global max pooling (X^{max}) operations are applied along the channel axes and, as in the *Channel Attention Block*, two distinct feature maps of size $N \times M \times 1$ are obtained. Instead of simply concatenating these two feature maps as in [204], they are combined by a weighted sum to enhance the network's learning capability. Subsequently, the combined feature map passes through a convolution layer with a filter size of 4×4 and the sigmoid activation function is applied, as reported in Equation (4.31).

$$A_s = \sigma(f^{4\times4}(\Lambda_2 \otimes X^{max} + \Gamma_2 \otimes X^{avg})) \tag{4.31}$$

where σ denotes the sigmoid function and $f^{4\times4}$ represents a convolution operation with the filter size of 4×4. It is worth noting that the filter size depends on the size of the areas that make up the city. For the case studies addressed in this work (see Section 4.6.2), which feature rather large regions, the proposed model does not need to focus on large area clusters; therefore, the size of the filter in this work (4×4) is reduced compared to those proposed in [204].

Finally, the process of getting spatial attention can be summarized as:

$$D^{''} = A_s \otimes D^{'} \tag{4.32}$$

If the reader is interested to explore the impact of the different components of the architecture, more details are presented in Appendix A.2

4.5 Related Prior Work

Several studies have addressed the problem of predicting vehicle flows in urban environments. This problem has been initially modeled as a time series prediction problem for each city area and approached through classical statistical methods at first, and ANN (e.g., deep learning) later. In particular, different statistical methods have been applied, including autoregressive integrated moving average (ARIMA) [129], Kalman filtering [71], and their variants, as well as other classical approaches such as Bayesian networks [172], Markov chain [146], and SVR models [207]. Other approaches have used k-means clustering, principal component analysis, and self-organizing maps to mine spatio-temporal performance trends [4]. However, classical statistics models show some weaknesses when applied to the flow prediction problem, namely they are unable to capture the spatial dependencies between the various areas because data for each region of the city are considered as independent time series, and they fail to capture the nonlinear relationship between space and time, which is essential for reliable prediction. Further studies overcame these downsides by considering spatial relationships [179] and external factors (e.g., environment and weather conditions [147]) within traditional time-series prediction methods.

ANN have exploited in flow predictions for their capability of capturing the non-linear spatial and temporal relationships within data. Initial works using ANN followed two main approaches. The first one exploits variants of RNN [7] such as *i)* LSTM [219] and *ii)* GRU [37], whose architectures can effectively capture both the long-term pattern and short-term fluctuation of time series. The second research line applies models based on CNN to identify spatial dependencies in traffic networks, treating dynamic traffic data as a sequence of frames [125]. Noticeably, while 2D convolutions with residual units [193, 223], 3D convolutions [34] and a combination of 2D and 3D convolutions [72] are widely used.

Those proposals, furthermore, do not feature specific architectural elements to capture temporal patters. Spatial and temporal dependencies are intrinsic to traffic data, making it essential to consider both aspects at the same time when predicting mobility dynamics. In this direction, deep learning-based approaches have been recently proposed, which exploit architectures able to capture spatial and temporal patterns. For this reason, authors in [119, 215] have combined convolution layers and LSTMs to capture both aspects: compared to models using only convolutions, they try to strengthen the model's ability to identify temporal patterns. Additionally, in [119, 215] an attention mechanism is used and in [212] simultaneously implements an autoencoder model with inner CMU layers.

In recent years, with the development of graph convolutional networks [99], which can be used to capture the structural characteristics of the graph network, we are witnessing their use in the field of traffic prediction [93]. Those approaches assume the existence of an origin-destination matrix, which provides details about the connections between different areas. This information, however, is often not available. One of the first graph-based works is [114] where the authors propose DCRNN, it is a model that captures the characteristic space through random walks on the graphs, and the temporal feature through the encoder-decoder architecture, while in [225] they apply the temporal graph convolutional network (T-GCN) model, which is in combination with the graph convolutional network (GCN) and gated recurrent unit (GRU). In [142] the authors propose a method of forecasting the traffic flow based on dynamic graphs: the traffic network is modeled by dynamic probability graphs. The convolution of the graph is performed on the dynamic graphs to learn the spatial features, which are then combined with the LSTM units to learn the temporal features. Finally, in [115] the authors propose

a dynamic perceptual graph neural network model for the temporal and spatial hidden relationships of deep learning segments. Indeed, the proposed model learns potential relationships of temporal features and spatial features. For a comprehensive review, we refer the reader to [109, 216], while a comprehensive library providing an open-source implementation of a number of models for traffic problems is presented in [194]. We provide a more detailed description of a selection of the approaches mentioned above in Section 4.6.

4.6 Experimental Analysis

This section reports an extensive experimental evaluation of the proposed models (3D-CLoST and STREED-Net) by comparing them with various reference models (as outlined in Section 4.6.1). The evaluation is conducted on three different case studies (described in detail in Section 4.6.2) using three different performance metrics. A computational complexity analysis is also included in the section.

4.6.1 Reference Methods

The proposed models are compared against the following state-of-the-art methods expressly devised to solve the citywide vehicle flow prediction problem [34]:

ST-ResNet [223]: it is one of the first deep learning approaches to traffic prediction. It predicts the flow of crowds in and out of each individual region of activity. ST-ResNet uses three residual networks that model the temporal aspects of *proximity*, *period*, and *trend* separately.

MST3D [34]: this model is architecturally similar to ST-ResNet. The three time dependencies and the external factors are independently modeled and dynamically merged by assigning different weights to different branches to obtain the new forecast. Differently from ST-ResNet, MST3D learns to identify space-time correlations using 3D convolutions.

ST-3DNet [72]: the network uses two distinct branches to model the temporal components of *closeness* and *trend*, while the daily period is left out. Both branches start with a series of 3D convolutional layers used to capture the spatio-temporal dependencies among the input frames. In the *closeness* branch, the output of the last convolutional layer is linked to a sequence of residual units to further investigate the spatial dependencies between the frames of the *closeness* period. The most innovative architectural element is the *Recalibration Block*. It is a block inserted at the end of each of the two main branches to explicitly model the contribution that each region makes to the prediction.

STAR [193]: this approach aims to model temporal dependencies by extracting representative frames of *proximity*, *period* and *trend*. However, unlike other solutions, the structure of the model consists of a single branch: the frames selected for the prediction are concatenated along the axis of the channels to form the main input to the network. In STAR as well, there is a sub-network dedicated to external factors and the output it generates is immediately added to the main network input. Residual learning is used to train the deep network to derive the detailed outcome for the expected scenarios throughout the city.

PredCNN [212]: this network builds on the core idea of recurring models, where previous states in the network have more transition operations than future states. PredCNN employs an autoencoder with CMU, which proved to be a valid alternative to RNN. Unlike the models discussed above, this approach considers only the temporal component of *closeness* but has a relatively complex architecture. The key idea of PredCNN is to sequentially capture spatial and temporal dependencies using CMU blocks.

ACFM [119]: this module is composed of two progressive Convolutional Long Short-Term Memory (ConvLSTM [164]) units connected via a convolutional layer. Specifically, the first ConvLSTM unit takes the

sequential flow features as input and generates a hidden state at each time-step, which is further fed into the connected convolutional layer for spatial attention map inference. The second ConvLSTM unit aims at learning the dynamic spatial-temporal representations from the attentionally weighted traffic flow features.

HA: the algorithm generates Inflow and Outflow forecasts by performing the arithmetic average of the corresponding values of the same day of the week at the same time as the instant in time to be predicted. This classical method represents a baseline in our comparative analysis, as it has not been developed specifically for the flow prediction problem.

Excluding MST3D, which has been entirely reimplemented following the indications of the original paper strictly, and PredCNN, whose original code has been completed of some missing parts, for all the other models the implementation released by the original authors has been used. The STREED-Net code, together with all the code realized for this research work, is freely available on GitHub [58].

We conclude this section by pointing out that, although the literature offers numerous proposals for deep learning models based on graphs with performances often superior to those of convolutional models, for the problem addressed in this paper, preliminary experiments that we conducted with graph-based models did not lead to satisfactory results. This is due to the nature of the problem considered, whose basic assumption is to be able to observe only the inflow and outflow across all areas of the city. Such a scenario is feasible and more realistic than one in which the trajectory or origin-destination pair of all vehicles is known but makes it impossible to create graphs with nontrivial connections (i.e., not between adjacent areas) for the problem under consideration.

4.6.2 Case Studies

Three real-life case studies are considered for the experimental analysis, which differ in both the city considered (New York and Beijing) and the type of vehicle considered (bicycle and taxi). This choice allow the models to be assessed on usage patterns that are expected to be significantly distinct. Follows a brief description of the considered case studies:

BikeNYC. In this first case study, the behavior of bicycles in New York City is analyzed. The data has been collected by the *NYC Bike system* in 2014, from 1 April to 30 September. Records from the last 10 days form the testing dataset, while the rest is used for training. The length of each time period is of 1h.

TaxiBJ. In the second case study, a fleet of cabs and the city of Beijing are considered. Data have been collected in 4 different time periods: 1 July 2013–30 October 2013, 1 March 2014–30 June 2014, 1 March 2015–30 June 2015, 1 November 2015–15 April 2016. The last four weeks are test data and the others are used for training purposes. The length of each time period is set to 30 min.

TaxiNYC. Finally, a dataset containing data from a fleet of taxicabs in New York is considered. Data have been collected from 1 January 2009 to 31 December 2014. The last four weeks are test data and the others are used for training purposes. The length of each time period is set to one hour. This case study has been specifically created to perform a more thorough and sound experimental assessments than those presented in the literature.

The city of New York has been tessellated into 16×8 regions, while the city of Beijing has been divided into 32×32 areas; the discrepancy in the number of regions considered is due to the large difference in extension between the two cities. The Beijing area (16,800 km^2) is 22 times bigger than the New York area (781 km^2).

The Beijing taxi dataset (*TaxiBJ*) and New York Bike dataset (*BikeNYC*) are available via [223]; they are already structured to carry out the experiments reported in this work. The TaxiNYC dataset has been specifically built for this research by processing and structuring data from the *NYC government website* [136].

A Min-Max normalization has been applied to all datasets to convert traffic values based on the scale $[-1,1]$. Note, however, that in the experiments a denormalization is applied to the expected values to be used in the evaluation.

In the three experiments, public holidays, metadata (i.e., DayOfWeek, Weekday/ Weekend) and weather have been considered as external factors. Specifically, the meteorological information reports the temperature, the wind speed, and the specific atmospheric situation (viz., sun, rain and snow).

4.6.3 Experimental Results

This section presents and discusses the results of experiments performed by running STREED-Net and the models presented in Section 4.6.1 on the three case studies. Moreover, three different evaluation metrics are used in this study to compare the results obtained: *RMSE*, *Mean Absolute Percentage Error (MAPE)* and *Absolute Percentage Error (APE)*, which are defined as follows:

$$RMSE = \sqrt{\frac{\sum_{n=1}^{N}\sum_{m=1}^{M}[(\hat{\imath}_{n,m}-\imath_{n,m})^2+(\hat{\omega}_{n,m}-\omega_{n,m})^2]}{N\times M}} \tag{4.33}$$

$$MAPE = 100\cdot\frac{\sum_{n=1}^{N}\sum_{m=1}^{M}\left|\frac{(\hat{\imath}_{n,m}-\imath_{n,m})+(\hat{\omega}_{n,m}-\omega_{n,m})}{(\hat{\imath}_{n,m}-\imath_{n,m})}\right|}{N\times M} \tag{4.34}$$

$$APE = 100\cdot\sum_{n=1}^{N}\sum_{m=1}^{M}\left|\frac{(\hat{\imath}_{n,m}-\imath_{n,m})+(\hat{\omega}_{n,m}-\omega_{n,m})}{(\hat{\imath}_{n,m}-\imath_{n,m})}\right| \tag{4.35}$$

where $\hat{\imath}_{n,m}$ and $\hat{\omega}_{n,m}$ are, respectively, the predicted Inflow and Outflow for region (n,m) at time t' and $N\times M$ is the total number of regions in the city.

It is worth noting that to account for and reduce the inherent stochasticity of learning-based models, each experiment was repeated ten times (replicas) using a different random seed in each replica. Mean and standard deviation are reported for each metric to provide a robust indication of the overall behavior of the compared methods.

Finally, tables in this section report the best results in **boldface** and the second best results in underlined.

BikeNYC

For the *BikeNYC* case study, STREED-Net parameters have been set as follows. The number n of input frames has been set to 4, the number L of encoding and decoding blocks has been set to 2. This decision has been dictated by the size of the grid (16×8): setting L greater than 2 (for example 3) would result in an encoder output tensor of size $4\times 2\times 1\times C$, which would be too small to allow the CMU block to effectively capture the time dependencies in the section located between the encoder and decoder. After some preliminary tests, the number of convolutional filters has been set to 64 in the first layer of the encoder and in the subsequent blocks, while in the last layer it has been set equal to 16. In this way, the dimensionality of the input vector goes from $I\in\mathbb{R}^{4\times 16\times 8\times 2}$ to $O\in\mathbb{R}^{4\times 4\times 2\times 16}$ as the encoder output. Symmetrically, the convolutions within the decoder use 64 filters, except for the final layer which uses only 2 filters to generate the prediction of the Inflow and Outflow channels. The parameters corresponding to the dimensionality of the convolution kernel, to the batch size and to the learning rate, have been optimized with the *Bayesian optimization* technique [167].

The best result has been obtained with a kernel size of 3, a batch size of 16 and a learning rate of 0.0001. The number of epochs is set to 150.

In terms of parameters for 3D-CLoST, we define the input volume using the autocorrelation function applied to the dataset. After removing the weekly seasonality (Figure 4.11), a strong correlation is identified with *i*) the 3 hours preceding the hour to be predicted, and *ii*) the hour before and after it on the previous week. Two 3D convolutional layers are used for this experiment, this choice is made due to the small size of each 3D volume (8×16) and the kernel sizes is set to $(2,3,3)$. The number of filters in the 3D convolutions varies in the two levels; in the first level is 32, while in the second level is 64. After the 3D convolutional layers, a max pooling layer with a reduced size of $(1,2,2)$ is applied. This model features two layers of LSTM, each with 350 units. The batch size is set to 64, the number of training epochs to 150 and the learning rate to 0.001.

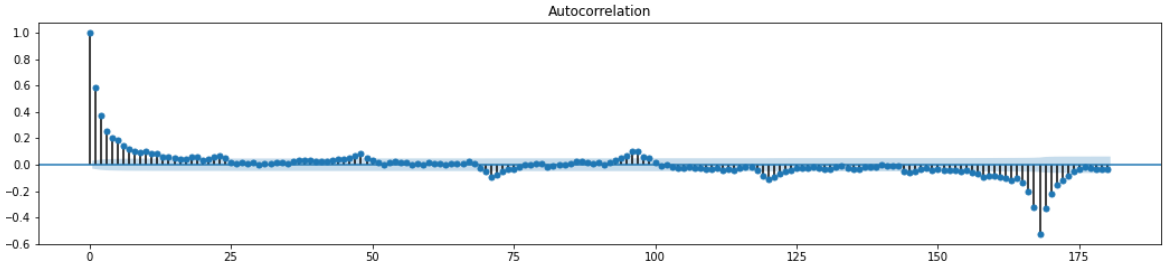


Figure 4.11 BikeNYC without weekly periodicity

Regarding the models from the literature, they have been arranged and trained following carefully the parameter values and indications reported in the respective publications.

As shown in Table 4.1, STREED-Net outperforms both 3D-CLoST and all other considered approaches in all evaluation metrics. In addition, the small standard deviation values are evidence of the robustness of the proposed approach. Nonetheless, it is worth observing that all learning-based approaches return similar results. We believe this is mainly due to the reduced size of the dataset that does not allow the models to be adequately trained. Moreover, the tessellation used in this case study (widely used in the literature), with a small grid of dimensions (16×8), tends to level off the metrics and hinder a more precise performance assessment.

Table 4.1 Results obtained for the Bike NYC data set.

Model	RMSE	MAPE	APE
HA	6.56	26.46	$4.09 \cdot 10^5$
ST-ResNet	5.01 ± 0.07	21.97 ± 0.26	$3.40 \cdot 10^5 \pm 4.06 \cdot 10^5$
MST3D	4.98 ± 0.05	22.03 ± 0.47	$3.41 \cdot 10^5 \pm 7.26 \cdot 10^5$
PredCNN	4.81 ± 0.04	21.38 ± 0.24	$3.31 \cdot 10^5 \pm 3.76 \cdot 10^5$
ST-3DNet	4.75 ± 0.06	21.42 ± 0.28	$3.31 \cdot 10^5 \pm 4.36 \cdot 10^5$
STAR	4.73 ± 0.05	20.97 ± 0.13	$3.24 \cdot 10^5 \pm 2.02 \cdot 10^5$
ACFM	4.68 ± 0.13	20.98 ± 0.68	$3.25 \cdot 10^5 \pm 1.05 \cdot 10^5$
3D-CLoST	4.90 ± 0.04	21.38 ± 0.20	$3.31 \cdot 10^5 \pm 3.12 \cdot 10^5$
STREED-Net	**4.67 ± 0.03**	**20.85 ± 0.15**	**$3.23 \cdot 10^5 \pm 2.31 \cdot 10^5$**

TaxiBJ

As with the experiment discussed above, for the *TaxiBJ* case study, the parameters of the models have been set according to the specifications given in the respective publications. In the case of STREED-Net, the hyperparameters are kept unchanged in the two experiments, except for the number L of encoding and decoding blocks, which has been increased to 3 because the grid is larger (32×32) in this experiment and more convolutional layers are needed to map the input tensor of the model. Also for this experiment, the kernel size, batch size, and learning rate parameters have been optimized with *Bayesian optimization* and the best values found were 3, 16, and 0.0001 respectively. The number of epochs has been set at 150. Notice that these values are the same used in the BikeNYC experiment.

In the 3D-CLoST model, only a few parameters have been modified from the previous experiment. Specifically, three layers of 3D convolutional layers are used. The choice has been made for the size of the input volume (32×32), which is much greater than the New York one. The number of filters on the three different levels of convolutional 3D have been set to 32, 64 and 64. The kernel size is set to (3, 3, 3). The model features two layers of LSTM, the first one has 500 units and the second has 250 units.

As can be seen from Table 4.2, STREED-Net outperforms 3D-CLoST and all other methods, in particular, reducing MAPE and APE by 2.9%, and 2.8%, respectively, compared with the second-best approach. The difference in performance in favor of the proposed model, in this experiment, is more appreciable because the dataset used for the training process is more significant but also because the number of regions is higher. This last consideration highlights how the proposed model seems suitable to be applied in real-world scenarios, i.e., where high model accuracy and dense tessellation are required (i.e., the city is partitioned into a large number of small regions).

Table 4.2 Results obtained for the Taxi Beijing dataset.

Model	RMSE	MAPE	APE
HA	40.93	30.96	$6.77 \cdot 10^7$
ST-ResNet	17.56 ± 0.91	15.74 ± 0.94	$3.45 \cdot 10^7 \pm 2.05 \cdot 10^6$
MST3D	21.34 ± 0.55	22.02 ± 1.40	$4.81 \cdot 10^7 \pm 3.03 \cdot 10^5$
PredCNN	17.42 ± 0.12	15.69 ± 0.17	$3.43 \cdot 10^7 \pm 3.76 \cdot 10^5$
ST-3DNet	17.29 ± 0.42	15.64 ± 0.52	$3.43 \cdot 10^7 \pm 1.13 \cdot 10^6$
STAR	16.25 ± 0.40	15.40 ± 0.62	$3.38 \cdot 10^7 \pm 1.36 \cdot 10^6$
ACFM	15.67 ± 0.23	15.16 ± 0.33	$3.32 \cdot 10^7 \pm 7.25 \cdot 10^5$
3D-CLoST	17.10 ± 0.23	16.22 ± 0.20	$3.55 \cdot 10^7 \pm 4.39 \cdot 10^5$
STREED-Net	**15.61 ± 0.11**	**14.73 ± 0.21**	**$3.22 \cdot 10^7 \pm 4.51 \cdot 10^5$**

TaxiNYC

As mentioned earlier, the TaxiNYC case study was created specifically to be able to evaluate the behavior of the proposed model in a wider set of scenarios than the literature. Consequently, in order to make a fair comparison, it was necessary to search for the best configuration of hyperparameters not only for the STREED-Net model but also for all the other approaches considered. The optimized parameters and the relative values used in the training phase are briefly summarized below for each model.

The unreported configuration values are the same as those used for the BikeNYC case study, since both experiments use the same map size (16×8). The parameters for each model are as follows:

- **ST-ResNet*.** Optimized parameters: number of residual units, batch size and learning rate. Optimal values found: 2, 16 and 0.0001.
- **MST3D.** Optimized parameters: batch size and learning rate. Optimal values found: 16 and 0.00034.
- **PredCNN.** Optimized parameters: encoder length, decoder length, number of hidden units, batch size and learning rate. Optimal values found: 2, 3, 64, 16 and 0.0001.
- **ST-3DNet.** Optimized parameters: number of residual units, batch size and learning rate. Best values found: 5, 16 and 0.00095.
- **STAR*.** Optimized parameters: number of remaining units, batch size and learning rate. Optimal values found: 2, 16 and 0.0001.
- **ACFM.** Optimized parameter: learning rate. Optimal value found: 0.0003.
- **3D-CLoST.** Optimized parameters: number of LSTM layers, number of hidden units in each LSTM layer, batch size, and learning rate. Optimal values found: 2, 500, 16, and 0.00076.
- **STREED-Net.** Optimized parameters: kernel size, batch size, and learning rate. Optimal values found: 3, 64 and 0.00086.

It is worth noting that, preliminary experiments showed a convergence issue for the training phase of both STAR and ST-ResNet models. In particular, they were unable to converge for any combination of parameters. This behavior is due to the strong presence of *outliers* and to the concentration of the relevant Inflow and Outflow values in a few central regions of the city. To overcome this issue, Batch Normalization layers have been inserted in the structure of the two models. In particular, Batch Normalization layers have been added after each convolution present in the residual units (a possibility that has already been foreseen in the original implementations) and after the terminal convolution of the networks (an option not considered in the source code provided by the original authors). For this reason, ST-ResNet and STAR are marked with an asterisk in the Table 4.3, which summarizes the experimental results.

Table 4.3 Results obtained for the Taxi New York dataset.

Model	RMSE	MAPE	APE
HA	164.31	27.19	$7.94 \cdot 10^5$
ST-ResNet*	**35.87 ± 0.60**	22.52 ± 3.43	$6.57 \cdot 10^5 \pm 1.00 \cdot 10^5$
MST3D	48.91 ± 1.98	23.98 ± 1.30	$6{,}98 \cdot 10^5 \pm 1.34 \cdot 10^4$
PredCNN	40.91 ± 0.51	25.65 ± 2.16	$7.49 \cdot 10^5 \pm 6.32 \cdot 10^4$
ST-3DNet	41.62 ± 3.44	25.75 ± 6.11	$7.52 \cdot 10^5 \pm 1.78 \cdot 10^5$
STAR*	36.44 ± 0.88	25.36 ± 5.24	$7.41 \cdot 10^5 \pm 1.53 \cdot 10^5$
ACFM	36.75 ± 0.94	**19.10 ± 1.08**	$\mathbf{5.58 \cdot 10^5 \pm 2.21 \cdot 10^4}$
3D-CLoST	48.17 ± 3.16	22.18 ± 1.05	$6.48 \cdot 10^5 \pm 3.08 \cdot 10^4$
STREED-Net	36.22 ± 0.72	20.29 ± 1.48	$5.93 \cdot 10^5 \pm 4.31 \cdot 10^4$

As it can be seen from Table 4.3, STREED-Net achieve excellent results in this experiment as well, ranked as one of the best models, while 3D-CLoST obtained poor performance ranking as the third-to-last model. In particular, as far as the RMSE is concerned, the performances obtained by STREED-Net are very close to the best one (achieved by ST-ResNet*, which is considerably different from the original ST-ResNet). As for MAPE and APE values, place our proposal ranks as the second-best approach, closely after ACFM.

4.6.4 Number of Trainable Parameters and FLOPs

A brief analysis of the number of trainable parameters and the computational complexity (measure in number of FLOPs) of each model for the different case studies is reported in this section. For what concerns the number of trainable parameters, as shown in Table 4.4, STREED-Net has a generally low number compared to other models as only STAR features lesser parameters to train. Such a reduced number of parameters is due to the fact that the dimensionality of the input is reduced by the encoder downsampling mechanism. The model with the highest number of parameters is 3D-CLoST, which uses both 3D convolutions and LSTM.

Table 4.4 Number of trainable parameters.

Model	BikeNYC	TaxiNYC	TaxiBJ
ST-ResNet	906,272	458,304	2,696,992
MST3D	668,218	668,378	8,674,370
PredCNN	3,967,906	3,967,906	4,827,842
ST-3DNET	540,696	617,586	903,242
STAR	161,052	310,076	476,388
ACFM	182,065	270,581	969,893
3D-CLoST	13,099,090	19,477,648	72,046,714
STREED-Net	582,673	582,673	765,497

Finally, Table 4.5 provide the computational complexity of each model in terms of floating point operations (FLOPs) as in [19] for each case study. As can be seen from the results obtained, the model with the higher computational complexity is PredCNN, which is based on CMUs. While, 3DCLoST is the model with the shortest forward and backward times. STREED-Net, instead, has a middle-range computational complexity compared to the other models, despite its autoencoder structure, the use of attention blocks, and CMUs. This occurs because although the number of network parameters is small, the network employs high complexity operators. However, the training and execution times of STREED-Net are compatible with its applicability in full-scale real-world scenarios.

Table 4.5 Computational complexity (in number of FLOPs).

Model	BikeNYC	TaxiNYC	TaxiBJ
ST-ResNet	230,849,450	115,735,786	5,459,663,018
MST3D	33,042,250	33,042,570	272,483,226
PredCNN	1,015,468,288	1,015,468,288	9,883,813,888
ST-3DNET	171,242,496	190,130,922	1,823,295,898
STAR	40,449,706	78,231,530	928,100,922
ACFM	41,687,924	93,643,568	621,498,864
3D-CLoST	29,613,094	9,601,920	338,148,804
STREED-Net	130,047,738	130,047,738	1,067,063,882

4.7 Case Study: Predicting Inflow and Outflow for Relocation

In this section we investigate the benefits of incorporating a Deep Learning model, in this case 3D-CLoST, into a relocation system to assess, in terms of satisfied demand and cost, the benefits of improving relocation

through the use of a predictive model. Specifically, we apply the model to forecast the number of e-scooters entering and leaving different areas of the city, allowing the relocation system to know in advance areas with a deficit or surplus of vehicles.

Identifying pick-up and drop-off zones

The relocation system receives as input the exact number of scooters needed to meet demand at a given time, as calculated by the $Delta(t,z)$ function, which represents the expected number of scooters added to an area z at a specific time t:

$$\Delta(t,z) = O(t,z) - D(t,z) - S(t,z) \tag{4.36}$$

where $O(t,z)$ is the number of vehicles expected to start a new trip from zone z in time $[t,t+\Delta T]$, where ΔT is one hour. $D(t,z)$, instead, is the predicted number of vehicles that will end a trip to such a zone in the same time interval. The difference between these two terms gives the predicted incoming or outgoing flow for a given zone at a given hour. $S(t,z)$ is the current number of e-scooters present in zone z. As such, if $\Delta(t,z) > 0$, zone z is expected to have a surplus - thus being a *pick-up* zone. Vice versa, if $\Delta(t,z) < 0$, zone z is expected to suffer from a lack of vehicles (*drop-off* zone), and $|\Delta(t,z)|$ represents how many scooters should be placed at zone z.

In this study, $O(t,z)$ and $D(t,z)$ are estimated in two different ways:

- **Baseline.** It is based on a simple stationary model for $O(t,z)$ and $D(t,z)$, where we assume that the average past demand is a good prediction of future demand as well. Therefore, we consider the type of day (weekday or weekend) and the time of day (in the 24 hours), obtaining 48-time slots in total. For each of them, the average number of e-scooters rented (returned), for each origin (destination) zone, is calculated. These averages are used to make predictions, that is, to obtain the $O(t,z)$ and $D(t,z)$ matrices. Then, we compute $Delta(t,z)$ with $S(t,z)$ as the current state.
- **DNN.** It refers to the 3D-CLoST model analyzed in Section 4.3. The outputs are the number of expected trips starting and ending in the next hour in each zone z, i.e., $O(t,z)$ and $D(t,z)$, from which we get $\Delta(t,z)$.

In our experimentation, we utilized e-scooter datasets from Austin [5] and Louisville [123]. We evaluated the performance of the two predictive models using the metric of RMSE and analyzed their impact on the relocation system.

Analysis of Predictions

Table 4.6 RMSE on the test

		Model RMSE	
		Baseline	DNN Driven
Austin	Origin $O(t,z)$	6.34	0.29
	Destination $D(t,z)$	6.20	0.29
Louisville	Origin $O(t,z)$	1.38	0.23
	Destination $D(t,z)$	1.32	0.22

Table 4.6 presents the RMSE values for the predictive models in Austin and Louisville. The DNN model demonstrates superior performance compared to the baseline model, with a significant improvement

in Austin. However, in Louisville, the difference in performance between the two models is less pronounced. This difference in performance may be ascribed to the smaller size and more homogenous usage patterns in Louisville compared to Austin. They refer to Appendix B for additional information.

4.8 Conclusion

Predicting vehicular flow is one of the central topics in the field of Smart Mobility. It is a challenging task, influenced by several complex factors, such as spatio-temporal dependencies and external factors. In this chapter, we have evaluated two Deep Learning proposals: 3D-CLoST and STREED-Net. The first model relies on multiple 3D CNNs and LSTM networks, while, the latter is based on 2D convolutions and CMU. Both methods are designed to predict the Inflow and Outflow in each region of the city. A comprehensive experimental campaign has been conducted on three different real-world case studies. According to the results, STREED-Net consistently outperforms 3D-CLoST and other state-of-the-art models in all the experiments conducted across the three performance metrics considered. We also analyze the results of a complexity analysis conducted on the architectures, revealing that 3D-CLoST has the lowest computational complexity.

Future Work. The integration of other external factors, such as the territorial characteristics of each geographical area, should be tested. Moreover, it would be appropriate to increase the granularity of the city tessellation, as well as conduct transfer learning experiments to study the applicability of the proposed models to scenarios with a reduced amount of data available.

5

Enhancing Spectral-Based GCNs to Solve Mobility Challenges

This chapter introduces SigMaNet, a generalized Graph Convolutional Network (GCN) capable of handling both undirected and directed graphs with weights not restricted in sign nor magnitude. The cornerstone of SigMaNet is the *Sign-Magnetic Laplacian* (L^{σ}), a new Laplacian matrix that we introduce *ex novo* in this work. L^{σ} allows us to bridge a gap in the current literature by extending the theory of spectral GCNs to (directed) graphs with both positive and negative weights. L^{σ} exhibits several desirable properties not enjoyed by other Laplacian matrices on which several state-of-the-art architectures are based, among which encoding the edge direction and weight in a clear and natural way that is not negatively affected by the weight magnitude. L^{σ} is also completely parameter-free, which is not the case of other Laplacian operators such as, e.g., the *Magnetic Laplacian*. The versatility and the performance of our proposed approach is amply demonstrated via computational experiments. Research reported in this chapter has been published in [63].

5.1 Introduction

The dramatic advancements of neural networks and deep learning have provided researchers and practitioners with extremely powerful analytics tools. Increasingly complex phenomena and processes which can often be modeled as graphs or networks, such as, e.g., social networks [8], knowledge graphs [231], protein interaction networks [97], or the World Wide Web (only to mention a few) can now be successfully addressed via Graph Convolutional Networks (GCNs).

Graph Convolutional Networks have also been used in the field of Smart Mobility [115, 217] because many occurrences in this area can be modeled as graphs, and Convolutional Neural Networks cannot be applied to non-Euclidean domains, as shown in Figure 5.1. Compared with other approaches, GCNs effectively manage to

represent the data and its interrelationships by explicitly capturing the topology of the underlying graph with a suitably-designed convolution operator. In the literature, GCNs mostly belong to two categories: spectral-based

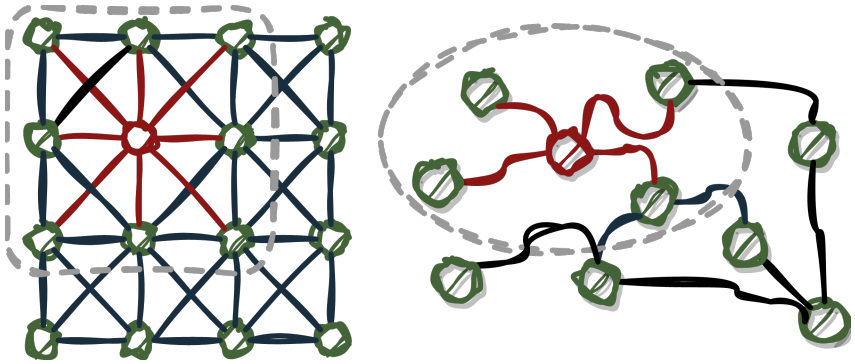


Figure 5.1 Illustration of 2D Convolutional Neural Networks (left) and Graph Convolotional Networks (right).

and spatial-based [208]. Spatial GCNs define the convolution operator as a localized aggregation operator [201] (although, rigorously speaking, such an operator may not always be called a convolution operator from a mathematical perspective). Differently, spectral GCNs define the convolution operator (a rigorous one in this case) as a function of the eigenvalue decomposition of the Laplacian matrix associated with the graph [99]. In their basic definition, both methods assume the graph to be undirected and to feature nonnegative weights. For a comprehensive review, we refer the reader to [77, 208].

Many real-world processes and phenomena can be modeled as directed graphs. While spatial GCNs have a natural extension to directed graphs, most spectral methods require the graph to be undirected and to feature nonnegative weights for their convolution operator to be well-defined. Indeed, due to being based on graph signal processing [35, 153], spectral GCNs require three fundamental properties to be satisfied which fail to hold when the graph is directed and/or features negative weights: *a)* the Laplacian matrix must be diagonalizable, i.e., it must admit an eigenvalue decomposition, *b)* the Laplacian matrix must be positive semidefinite, and *c)* the spectrum of the normalized Laplacian matrix must be upper-bounded by 2 [99, 208].

In recent years, several extensions of the definition of Laplacian matrix have been proposed to overcome the first limitation, i.e., to handle directed graphs (see, e.g., the *Magnetic Laplacian* [222, 224] and the *Approximate Digraph Laplacian* constructed via the PageRank matrix [180]). Differently, no spectral techniques have been introduced so far to overcome the second limitation, i.e., to handle graphs with edge weights unrestricted in sign, which arise in many relevant applications (such as, e.g., those where the graph models credit/debit transactions between customers, like/dislike evaluations among users, or positive/negative user opinions). Our proposal extends the use of spectral-based GCN to various mobility problems, including the prediction of electric cars charging and discharge. City locations form a directed graph, where vehicles charging or discharging can be determined based on the link direction (downhill for charging or flat/uphill for discharging). This graph is composed of both the direction and sign (positive/negative) of the edges.

The remainder of the Chapter is structured as follows. Preliminaries and previous works are summarized in Section 5.2. The *Sign-Magnetic Laplacian* operator is introduced in Section 5.3 together with its properties. The section also provides an overview of the SigMaNet architecture that we build upon the Sign-Magnetic Laplacian. Computational results are reported in Section 5.4, where we apply our model to node classification and link prediction tasks and compare it to different state-of-the art spectral and spatial methods. Finally, conclusions and recommendations for future work are discussed in Section 5.5. The proofs of our theorems and further numerical results are provided in Appendix C.

5.2 Preliminaries and previous works

For a given $n \in \mathbb{N}$, we denote by $[n]$ the set of integers $\{1,\dots,n\}$. For a given matrix M of appropriate dimensions with real eigenvalues, we denote its largest eigenvalue by $\lambda_{\max}(M)$. Throughout the paper, e and $ee^\top$ denote the all-one vector and matrix of appropriate dimensions. Undirected and directed graphs are denoted by $G=(V,E)$, where V is the set of vertices and E the set of edges. In the undirected (directed) case, E is a collection of unordered (ordered) pairs of elements in V. G is always assumed to be self-loop free.

5.2.1 Generalized convolution matrices

For some $n \in \mathbb{N}$, let $M \in \mathbb{C}^{n\times n}$ be a positive semidefinite Hermitian matrix[1] with eigenvalue (or spectral) decomposition $M = U\Lambda U^*$, where $\Lambda \in \mathbb{R}^{n\times n}$ is the diagonal matrix whose elements are the (real, as M is Hermitian) eigenvalues of M, $U \in \mathbb{C}^{n\times n}$, and U^* is the complex conjugate of U. For each $i \in [n]$, the i-th column of U coincides with the i-th eigenvector of M corresponding to its i-th eigenvalue Λ_{ii}. The columns of U form a basis of $\mathbb{C}^n$. We assume $\lambda_{\max}(M) \leq 2$.

Given a *signal* $x \in \mathbb{C}^n$, let $\widehat{x}$ be its discrete Fourier transform with basis U, i.e., $\widehat{x} = U^*x$. As $U^{-1} = U^*$, the transform is invertible and the inverse transform of $\widehat{x}$ reads $x = U\widehat{x}$. Given a *filter* $y \in \mathbb{C}^n$, the transform of its convolution with x satisfies the relationship $\widehat{y*x} = \widehat{y} \odot \widehat{x} = \mathrm{Diag}(\widehat{y})\widehat{x}$, where $*$ and $\odot$ denote the convolution and the Hadamard (or component-wise) product, respectively. Applying the inverse transform, we have $y*x = U\,\mathrm{Diag}(\widehat{y})U^*x$. Letting $\Sigma := \mathrm{Diag}(\widehat{y})$, we call a *generalized convolution matrix* the matrix $Y := U\Sigma U^*$, as $y*x = Yx$.

Let $\tilde{\Lambda} := \frac{2}{\lambda_{\max}(M)}\Lambda - I$ be the normalization of Λ. As $UU^* = I$, the same normalization applied to M leads to $\tilde{M} = U\tilde{\Lambda}U^* = \frac{2}{\lambda_{\max}}M - I$. Following [74, 99], we assume that y is such that the entries of $\widehat{y}$ are real-valued polynomials in $\tilde{\Lambda}$, i.e., that $\widehat{y}_i = \sum_{k=0}^K \theta_k T_k(\tilde{\lambda}_i), i \in [n]$, where $\theta_0,\dots,\theta_K \in \mathbb{R}$, $K \in \mathbb{N}$, and T_k is the Chebyshev polynomial of the first kind of order k. T_k is recursively defined as $T_0(x) = 1, T_1(x) = x$, and $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ for $k \geq 2$, with $x \in \mathbb{R} \cap [-1,1]$. Thus, we rewrite Σ as $\Sigma = \mathrm{Diag}(\widehat{y}) = \sum_{k=0}^K \theta_k T_k(\tilde{\Lambda})$, where T_k is applied component-wise to $\tilde{\Lambda}$, i.e., $(T_k(\tilde{\Lambda}))_{ij} = T_k(\tilde{\Lambda}_{ij})$ for all $i,j \in [n]$. With this, the convolution of x by y can be rewritten as $Yx = U\Sigma U^*x = U\left(\sum_{k=0}^K \theta_k T_k(\tilde{\Lambda})\right)U^*x$. Since, as it is easy to verify, $(U\tilde{\Lambda}U^*)^k = U\tilde{\Lambda}^k U^*$ holds for all $k \in \mathbb{N}$, one can also verify that $Yx = U\left(\sum_{k=0}^K \theta_k T_k(\tilde{\Lambda})\right)U^*x = \sum_{k=0}^K \theta_k T_k(U\tilde{\Lambda}U^*)x = \sum_{k=0}^K \theta_k T_k(\tilde{M})x$.

Assuming $\lambda_{\max} = 2$, we have $\tilde{M} = M - I$. Letting $K = 1$ and $\theta_1 = -\theta_0$, deduce:

$$y*x = Yx = (\theta_0 I - \theta_0(M-I))x = \theta_0(2I - M)x. \tag{5.1}$$

If M is chosen so as to express the topology of the graph and x coincides with the graph features, Equation (5.1) represents the convolution operation underlying a spectral GCN. M should satisfy three properties for Equation (5.1) to apply:

i) it should admit an eigenvalue decomposition,

ii) it should be positive semidefinite, and

iii) its spectrum should be upper-bounded by 2.

Examples of M are given in the following subsections.

[1] A matrix is called Hermitian if its real part is symmetric and its imaginary part skew-symmetric.

5.2.2 Spectral convolutions for undirected graphs

Let $G = (V, E)$ be an undirected graph with $n = |V|$ without weights nor signs associated with its edges and let $A \in \{0,1\}^{n \times n}$ be its adjacency matrix, with $A_{ij} = 1$ if and only if $\{i, j\} \in E$. The Laplacian matrix of G is defined as:

$$L := D - A,$$

where $D := \mathrm{Diag}(Ae)$ is a diagonal matrix and, for each $i \in V$, D_{ii} equal to the degree of node i [38]. The normalized Laplacian matrix is defined as:

$$L_{\mathrm{norm}} := D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}.$$

L_{norm} satisfies many properties, among which *i)*, *ii)* and *iii)*.

The spectral convolution on the undirected graph G introduced by [99] is obtained by letting $M := L_{\mathrm{norm}}$ and defining Y as done before. Equation (5.1) becomes:

$$\begin{aligned} y * x = Yx &= \theta_0(2I - (I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}})x \\ &= \theta_0(I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}})x. \end{aligned} \tag{5.2}$$

To alleviate numerical instabilities and exploding/vanishing gradients when training a GCN built on Equation (5.2), [99] suggest the adoption of the following modified equation with a modified convolution matrix $\tilde{Y}$:

$$y * x = \tilde{Y}x = \theta_0(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}})x, \tag{5.3}$$

where $\tilde{A} := A + I$ and $\tilde{D} := \mathrm{Diag}(\tilde{A}e)$.

Drawbacks The Laplacian matrix L is well-defined only for undirected graphs with (if any) nonnegative weights. The reason is twofold. *i)* If G is a directed graph, the adjacency matrix A (and, thus, L) is, in the general case, not symmetric and, thus, L may not admit an eigenvalue decomposition. *ii)* If G is directed, the sum of the rows of A is not necessarily identical to the sum of its columns, and, thus, the matrix D is not well-defined (as $\mathrm{Diag}(Ae) \neq \mathrm{Diag}(e^\top A)$). If, on the other hand, G is undirected but features edges $\{i, j\} \in E$ with a negative weight $w_{ij} < 0$, L is not positive semidefinite in the general case as, even if $\mathrm{Diag}(Ae) = \mathrm{Diag}(e^\top A)$, $D^{-\frac{1}{2}} \notin \mathbb{R}^{n \times n}$ if $D_{ii} < 0$ for some $i \in V$.

5.2.3 Extending spectral convolutions to directed graphs

Since the adjacency matrix of a directed graph is asymmetric, the Laplacian matrix L defined before does not enjoy properties *i)*, *ii)* and *iii)* and, therefore, it is not possible to directly apply Equation (5.2) to define a spectral graph convolution. Alternative approaches such as those of [181] and [180] (which split the adjacency matrix A into a collection of symmetric matrices in such a way that the information regarding the direction of the edges is not lost) are known, but they typically come at the cost of increasing the size and complexity of the neural network.

A more direct way to encode the directional information of the edges is resorting to complex-valued matrices that are Hermitian. Indeed, albeit asymmetric in the general case, Hermitian matrices admit an eigenvalue decomposition with real eigenvalues. The *Magnetic Laplacian* is one such matrix. It was first introduced in

particle physics and quantum mechanics by [117] and then applied in the context of community detection by [54], in graph signal processing by [66] and, lastly, in the context of spectral GCNs by [222, 224].

Let $A_s := \frac{1}{2}\left(A + A^\top\right)$ be the symmetrized version of A and let $D_s := \mathrm{Diag}(A_s e)$. The *Magnetic Laplacian* is defined as the following Hermitian positive semidefinite matrix:

$$L^{(q)} := D_s - H^{(q)},$$

with

$$H^{(q)} := A_s \odot \exp\left(\mathbf{i}\Theta^{(q)}\right), \Theta^{(q)} := 2\pi q\left(A - A^\top\right),$$

where $\mathbf{i}$ is the square root of the negative unit, i.e., $\mathbf{i} = \sqrt{-1}$, and $\exp\left(\mathbf{i}\Theta^{(q)}\right) = \cos(\Theta^{(q)}) + \mathbf{i}\sin(\Theta^{(q)})$, where $\cos(\cdot)$ and $\sin(\cdot)$ are applied component-wise. Θ is a phase matrix that captures the directional information of the edges. The parameter $q \geq 0$ represents the electric charge. It is typically set to values smaller than 1 such as $\left[0, \frac{1}{4}\right]$ as in [224] or $\left[0, \frac{1}{2}\right]$ as in [55]. If $q = 0$, $\Theta^{(q)} = 0$ and $L^{(q)}$ boils down to the Laplacian matrix L defined on the "symmetrized" version of the graph with adjacency matrix A_s (which, crucially, renders G completely undirected and its directional information is lost).

For unweighted directed graphs where $A \in \{0,1\}^{n\times n}$, $H^{(q)}$ straightforwardly captures the graph's directional information. Assuming $q = 0.25$, we have $H^{(q)}_{ij} = H^{(q)}_{ji} = 1 + \mathbf{i}0$ if $(i,j),(j,i) \in E$ and $H^{(q)}_{ij} = 0 + \mathbf{i}\frac{1}{2}$ and $H^{(q)}_{ji} = 0 - \mathbf{i}\frac{1}{2}$ if $(i,j) \in E \wedge (j,i) \notin E$. This way, digons (pairs of antiparallel edges) are represented as single undirected edges in the real part of $H^{(q)}$ whereas any other edge is represented in the imaginary part of $H^{(q)}$ with a sign encoding its direction.

Drawbacks The *Magnetic Laplacian* $L^{(q)}$ suffers from two drawbacks. The first Drawback is that $L^{(q)}$ is well-defined only for graphs with nonnegative weights. Indeed, if $(D_s)_{ii} < 0$ for some $i \in V$, in the general case $L^{(q)}$ is not positive semidefinite and $D_s^{-\frac{1}{2}}$ does not belong to $\mathbb{R}^{n\times n}$. The second Drawback is that, even when restricted to graphs with nonnegative weights, $L^{(q)}$ exhibits a crucial sign-pattern inconsistency if the edge weights are sufficiently large. Indeed, while for unweighted graphs $L^{(q)}$ always captures the directional information of the edges by the sign of the imaginary part of $H^{(q)}$, this (as we are about to show) is not necessarily the case for weighted graphs, where the sign pattern of $H^{(q)}$ can drastically change irrespective of the edge direction by just scaling the edge weights by a positive constant. To see this, assume, for instance, $(i,j) \in E$ and $(j,i) \notin E$ with $A_{ij} = 1$. Then, we obtain: $H^{(0.25)}_{ij} = 0.40 \cdot 0.31 + \mathbf{i}0.40 \cdot 0.95$ and $H^{(0.25)}_{ji} = 0.40 \cdot 0.31 - \mathbf{i}0.40 \cdot 0.95$ by scaling A_{ij} by 0.8; $H^{(0.25)}_{ij} = -1 + \mathbf{i}0$ by scaling A_{ij} by 2; $H^{(0.25)}_{ij} = 0 + \mathbf{i}\frac{5}{2}$ by scaling A_{ij} by 5; and $H^{(0.25)}_{ij} = \frac{36}{2} + \mathbf{i}0$ by scaling A_{ij} by 36. This shows that $L^{(q)}$ is not robust to scaling and that, in it, the edge direction information can easily be lost. A full example of this behavior is reported in Appendix C.2.

5.3 Our proposal: the Sign-Magnetic Laplacian and SigMaNet

In this section, we extend the theory underlying spectral GCNs by introducing the *Sign-Magnetic Laplacian* matrix, a positive semidefinite Hermitian matrix that well captures the directional as well as the weight information of any directed graph with weights unrestricted in sign nor magnitude without suffering from the two drawbacks we outlined before.

Detailed proofs of the theorems contained in this section are provided in Appendix C.1.

Sign-Magnetic Laplacian

We introduce the following Hermitian matrix, which we refer to as the *Sign-Magnetic Laplacian*:

$$L^{\sigma} := \bar{D}_s - H^{\sigma},$$

with

$$H^{\sigma} := A_s \odot \left(ee^{\top} - \operatorname{sgn}(|A - A^{\top}|) + \mathbf{i} \operatorname{sgn}\left(|A| - |A^{\top}|\right) \right),$$

where $A_s := \frac{1}{2}\left(A + A^{\top}\right)$, $\bar{D}_s := \operatorname{Diag}(|A_s|\, e)$, and $\operatorname{sgn} : \mathbb{R} \to \{-1, 0, 1\}$ is the signum function (applied component-wise).

Let us illustrate the way the graph topology and its weights are stored in H^{σ}. H^{σ} encodes the direction and weight of every edge $(i, j) \in E$ that does not have an antiparallel edge (j, i) purely in its imaginary part by $H^{\sigma}_{ij} = -H^{\sigma}_{ji} = 0 + \mathbf{i}\frac{1}{2}A_{ij}$. Pairs of antiparallel edges with $A_{ij} = A_{ji}$ are encoded purely in the real part by $H^{\sigma}_{ij} = H^{\sigma}_{ji} = \frac{1}{2}(A_{ij} + A_{ji}) + \mathbf{i}0$ (as if they coincided with an undirected edge of the same weight). Differently, pairs of antiparallel edges with $A_{ij} \neq A_{ji}$ are encoded purely in the imaginary part by $H^{\sigma}_{ij} = -H^{\sigma}_{ji} = 0 + \mathbf{i}\frac{1}{2}(A_{ij} + A_{ji})$ if $|A_{ij}| > |A_{ji}|$ and $H^{\sigma}_{ij} = -H^{\sigma}_{ji} = 0 - \mathbf{i}\frac{1}{2}(A_{ij} + A_{ji})$ if $|A_{ij}| < |A_{ji}|$.

We define the normalized version of L^{σ} as:

$$L^{\sigma}_{\text{norm}} := \bar{D}_s^{-\frac{1}{2}} L^{\sigma} \bar{D}_s^{-\frac{1}{2}} = I - \bar{D}_s^{-\frac{1}{2}} H^{\sigma} \bar{D}_s^{-\frac{1}{2}}. \tag{5.4}$$

One can show that both L^{σ} and L^{σ}_{norm} are Hermitian by construction. Therefore, they admit an eigenvalue decomposition and, thus, satisfy property *i)*.

L^{σ} is defined in such a way that, if G is unweighted, it mirrors the behavior of $L^{(q)}$ with $q = 0.25$:

Theorem 1. *If $A \in \{0, 1\}^{n \times n}$ and $q = 0.25$, $L^{\sigma} = L^{(q)}$.*

In contrast with $L^{(q)}$, L^{σ} does not suffer from drawback #1 as it is well-defined even when G features negative weights.

With the following two results, we show that L^{σ} and L^{σ}_{norm} enjoy the two remaining properties *ii)* and *iii)* that are required for the construction of a convolution operator:

Theorem 2. *L^{σ} and L^{σ}_{norm} are positive semidefinite.*

Theorem 3. *$\lambda_{max}(L^{\sigma}_{norm}) \leq 2$.*

With the next result, we show that L^{σ} encodes the topology of G (including its directions) and the weights of its edges in such a way that it is always proportional to the magnitude of A (i.e., to the magnitude of graph weights):

Theorem 4. *Given a constant $\alpha \in \mathbb{R}^{+}$, L^{σ} satisfies the following positive homogeneity property:*

$$L^{\sigma}(\alpha A) = \alpha L^{\sigma}(A),$$

where $L^{\sigma}(\alpha A)$ and $L^{\sigma}(A)$ are the Sign-Magnetic Laplacian matrices of a directed graph with, respectively, adjacency matrix $\alpha A \in \mathbb{R}^{n \times n}$ and $A \in \mathbb{R}^{n \times n}$.

Theorem 4 shows that L^{σ} is robust to scaling applied to the weights of G. From it, we deduce the following result, which shows that L^{σ} does not suffer from drawback #2:

Corollary 1. *The sign-pattern of L^σ is uniquely determined by the topology of G and, thus, L^σ does not suffer from any sign-pattern inconsistencies.*

Lastly, we show that L^σ satisfies the following invariant:

Theorem 5. *Consider a weighted digon-free directed graph $G = (V,E)$. Given a directed edge $(i,j) \in E$ of weight w_{ij}, let $G' = (V,E')$ be a graph obtained by reversing the direction of (i,j) in G into (j,i) and flipping the sign of its weight by letting $w_{ji} = -w_{ij}$. Let $L^\sigma(G)$ and $L^\sigma(G')$ be the L^σ matrix defined on G and G', respectively. Then:*

$$L^\sigma(G) = L^\sigma(G').$$

Theorem 5 shows that the behavior of L^σ is consistent with applications where the graph models a flow relationship in which flipping the sign of an edge coincides with flipping its direction. This applies to, among others, scenarios where the weights represent flow values such as cash flows, where it is reasonable to assume that a negative flow from i to j corresponds to a positive flow from j to i. Further details on the impact of this property are reported in Appendix C.3.

5.3.1 SigMaNet's architecture

As previously discussed, for the spectral convolution operator to be well defined the Laplacian matrix must satisfy properties *i)*, *ii)*, and *iii)*. As the hermiticity of L^σ_{norm}, Theorem 2, and Theorem 3 show that L^σ_{norm} enjoys these properties, Equation (5.2) can be rewritten as:

$$Yx = \theta_0 \left(I + \bar{D}_s^{-\frac{1}{2}} H^\sigma \bar{D}_s^{-\frac{1}{2}} \right) x.$$

Following [99] to avoid numerical instabilities, we apply Equation (5.3) with $\tilde{D}_s^{-\frac{1}{2}} \tilde{H}^\sigma \tilde{D}_s^{-\frac{1}{2}}$ *in lieu* of $I + \bar{D}_s^{-\frac{1}{2}} H^\sigma \bar{D}_s^{-\frac{1}{2}}$, where $\tilde{H}^\sigma$ and $\tilde{D}_s$ are defined based on $\tilde{A} := A + I$ rather than A. We generalize the feature vector signal $x \in \mathbb{C}^{n\times 1}$ to a feature matrix signal $X \in \mathbb{C}^{n\times c}$ with c input channels (i.e., a c-dimensional feature vector for every node of the graph). Letting $\Theta \in \mathbb{C}^{c\times f}$ be a matrix of learnable filter parameters with f filters and ϕ be an activation function applied component-wise to the input matrix, the output $Z^\sigma \in \mathbb{C}^{n\times f}$ of SigMaNet's convolutional layer is:

$$Z^\sigma(X) = \phi(\tilde{D}_s^{-\frac{1}{2}} \tilde{H}^\sigma \tilde{D}_s^{-\frac{1}{2}} X\Theta). \tag{5.5}$$

Since the argument of ϕ is a complex matrix and, thus, traditional activation functions cannot be directly adopted, we follow [224] and rely on a complex version of the *ReLU* activation function which is defined for a given $z \in \mathbb{C}$ as $\phi(z) = z$ if $\Re(z) \geq 0$ and $\phi(z) = 0$ otherwise. As the output of the convolutional layer Z^σ is complex-valued, to coerce it into the reals without information loss we apply an *unwind* operation by which $Z^\sigma(X) \in \mathbb{C}^{n\times f}$ is transformed into $[\Re(Z^\sigma(X)); \Im(Z^\sigma(X)] \in \mathbb{R}^{n\times 2f}$. To obtain the final result based on the task at hand, we apply either a linear layer with weights W or a $1D$ convolution.

Considering, e.g., the task of predicting the class of an edge, SigMaNet is defined as:

$$\text{softmax}\left(\text{unwind}\left(Z^{\sigma(2)}\left(Z^{\sigma(1)}\left(X^{(0)}\right)\right)\right)W\right),$$

where $X^{(0)} \in \mathbb{R}^{n\times c}$ is the input feature matrix, $Z^{\sigma(1)} \in \mathbb{C}^{n\times f_1}$ and $Z^{\sigma(2)} \in \mathbb{C}^{n\times f_2}$ are the spectral graph convolutional layers, $W \in \mathbb{R}^{2f_2\times d}$ are the weights of the linear layer (with d being the number of classes), and $\text{softmax} : \mathbb{R}^d \to \mathbb{R}^d$ is the normalized exponential activation function.

SigMaNet features a flexible architecture that differs from other spectral GCNs proposed in the literature (e.g., MagNet [224]) mainly in the way the convolutional layer is defined. As such, it can easily be applied to a variety of tasks in an almost task-agnostic way (provided that one defines a suitable loss function). As L^{σ} is entirely parameter-free, SigMaNet does not require any fine-tunings to optimize the propagation of topological information through the network, differently from, e.g., DiGraph [180] and MagNet [224].

5.3.2 Complexity of SigMaNet

Assuming, as done in our experiments, that SigMaNet features two graph-convolutional layers with f_1 and f_2 filters, each defined as in Equation (5.5) and c features per node, the complexity of SigMaNet is $O(nc(n+f_1)+nf_1(n+f_2)+\Gamma)$, where Γ accounts for the complexity of the last (task-specific) layer. For the four tasks that we consider in the next section, we have $\Gamma = m^{\text{train}} f_2 d$ for the first three (link sign/direction/existence prediction), where m^{train} is the number of edges in the training set and d is the number of classes, and $\Gamma = nf_2 d$ for the last one (node classification). Such a complexity is quadratic in n and, assuming $O(f_1) = O(f_2) = O(c) = O(d)$, also quadratic in the dimension of the feature/class space. Detailed calculations are reported in Appendix C.6.

We remark that, while enjoying a wider applicability due to being able to handle graphs with edge weights unrestricted in sign), SigMaNet features half the weights of MagNet.

5.4 Experimental Analysis

In this section, we report on a set of computational experiments carried out on four tasks: *link sign prediction*, *link existence prediction*, *link direction prediction*, and *node classification*. The experiments are conducted to assess the performance of SigMaNet on graphs with weights unrestricted in sign on which no other spectral GCNs can be applied (link sign prediction) and to compare it to other state-of-the-art spectral and spatial approaches on graphs with nonnegative weights (link existence/direction prediction and node classification). The code is available on GitHub [56].

For the link sign prediction task, we compare SigMaNet with three categories of methods: *i)* signed network embedding: SiNE [198], SIGNet [88], BESIDE [36]; *ii)* Feature Engineering: FeExtra [110]; and *iii)* signed Graph Neural Networks: SGCN [47], SiGAT [81], and SDGNN [82]. For the link prediction and node classification tasks, we compare SigMaNet with the following three categories of methods: *i)* spectral methods designed for undirected graph: ChebNet [45], GCN [99]; *ii)* spectral methods designed for directed graphs: DGCN [181], DiGraph [180], DiGCL [182], and MagNet [224], and *iii)* spatial methods: APPNP [100], SAGE [73], GIN [211], GAT [191], and SSSNET [76].

Throughout the tables contained in this section, the best results are reported in **boldface** and the second best are underlined. Further results and analysis are reported in Appendix C.5.

5.4.1 Datasets

We test SigMaNet on six real-world datasets from the literature: `Bitcoin-OTC` and `Bitcoin Alpha` [106]; `Slashdot` and `Epinions` [111]; `WikiRfa` [203]; and `Telegram` [26]. In order to better assess SigMaNet's performance as the density of the graph increases, in three tasks we also consider a synthetic set of graphs generated via a direct stochastic block model Direct Stochastic Block Model (DSBM) with (unlike what is done in [224]) edge weights greater than 1. These datasets are generated by varying: *i)* the number of nodes n; *ii)* the number of clusters C; *iii)* the probability α_{ij} to create an undirected edge between nodes i and j

belonging to different clusters ; *iv)* the probability α_{ii} to create an undirected edge between two nodes in the same cluster, and *v)* the probability β_{ij} of an edge taking a certain direction. Each node is labeled with the index of the cluster it belongs to. A more detailed description of the datasets we use can be found in Appendix C.4.

5.4.2 Link sign prediction

The link sign prediction task is a classification problem designed for graphs with both positive and negative edge weights. It consists of predicting the sign of the edges in the graph and, thus, for such task SigMaNet is the only spectral-based GCN that can be used.

For this task, we adopt the `Bitcoin Alpha`, `Bitcoin-OTC`, `WikiRfa`, `Slashdot`, and `Epinions` datasets, which are directed graphs with weights of unrestricted sign (necessary for the task to be applicable) and of arbitrary magnitude, with the sole exception of the last two, whose weights satisfy $A \in \{-1,0,+1\}^{n\times n}$. In these five datasets, the classes of positive and negative weighted edges are imbalanced (i.e., nearly 80% are positive edges). The experiments are run with k-cross validation with $k = 5$, reporting the average score obtained across the k splits. Connectivity is maintained when building each training set by guaranteeing that the graph used for training in each fold contains a spanning tree. Following [82], we remove 20% of the edges for testing and use the remaining 80% for training.

The results are reported in Table 5.1.[2] We observe that SigMaNet clearly outperforms all competitors on the three datasets whose graphs have unrestricted weights, i.e., `Bitcoin Alpha`, `Bitcoin-OTC`, and `WikiRfa`. On graphs with unit weights, i.e., `Slashdot` and `Epinions`, its performance, while marginally worse, is still in line with the best methods. This suggests the relevance of Corollary 1 towards SigMaNet's performance. We remark that the latter is achieved in spite of SigMaNet being less complex than the deep neural networks we compared it to here, which feature two sequentially-applied neural networks (one producing a set of embeddings from which the other one predicts the link sign via a logistics regression).

5.4.3 Link (existence and direction) prediction

We now consider two tasks: *existence prediction* and *direction prediction*. In the first one, the model is asked to predict whether $(u,v) \in E$ for a given pair of vertices $u,v \in V, u \neq v$ provided as input. The second one is a binary task where the model is asked to predict whether *a)* $(u,v) \in E$ or *b)* $(v,u) \in E$ or both.

For both tasks, we only consider graphs with nonnegative edge weights. This allows us to compare SigMaNet not just to spatial GCNs as done before, but also to state-of-the-art spectral-based ones. As such GCNs are designed solely for graphs with nonnegative weights, one may expect that the wider applicability of SigMaNet should come at the price of inferior performances. Our experiments show that this is not the case.

The datasets that we consider are: `Telegram`, `Bitcoin Alpha`, `Bitcoin-OTC`, and synthetic DBSM graphs. The latter are generated with $n = 2500$, $C = 5$, $\alpha_{ii} = 0.1$, $\beta_{ij} = 0.2$, with an increasing inter-cluster density $\alpha_{ij} \in \{0.05, 0.08, 0.1\}$. [3] Following [224], in each task we reserve 15% of the edges for testing, 5% for validation, and use the remaining ones for training. The experiments are run with k-cross validation with $k = 10$, preserving graph connectivity.

[2] Except for SigMaNet, the results are taken from [82]. For SGCN, SiGAT, and SDGNN, we chose to report the results in [82] rather than those in [77] as the former are better, and, thus, more challenging for SigMaNet.

[3] As spectral methods, except for SigMaNet, cannot handle graphs with negative weights, to be able to compare our proposal to them in a setting in which the latter can be applied, in these experiments we pre-process `Bitcoin-OTC` and `Bitcoin Alpha` by removing any edge with a negative weight—in the tables, these datasets are denoted by a '*'.

Table 5.1 Link sign prediction results assessed with four metrics

		Signed Network Embedding			Feature Engineering	Graph Neural Network			
Dataset	Metric (%)	SiNE	SIGNet	BESIDE	FeExtra	SGCN	SiGAT	SDGNN	SigMaNet
Bitcoin Alpha	Micro-F1	94.58	94.22	94.89	94.86	92.56	94.56	94.91	**95.13**
	Binary-F1	97.16	96.96	97.32	97.30	96.07	97.14	97.29	**97.44**
	Macro-F1	68.69	69.65	73.00	71.67	63.67	70.26	73.90	**74.69**
	AUC	87.28	89.08	89.81	88.82	84.69	88.72	89.88	**92.46**
Bitcoin-OTC	Micro-F1	90.95	92.29	93.20	93.61	90.78	92.68	93.57	**94.49**
	Binary-F1	95.10	95.81	96.28	96.53	94.91	96.02	96.47	**97.02**
	Macro-F1	68.05	73.86	78.43	78.26	73.06	75.33	80.17	**80.53**
	AUC	85.71	89.35	91.52	91.21	87.55	90.55	91.24	**93.67**
WikiRfa	Micro-F1	83.38	83.84	85.89	83.46	84.89	84.57	86.27	**86.56**
	Binary-F1	89.72	90.01	91.17	89.87	90.69	90.42	91.42	**91.64**
	Macro-F1	73.19	73.84	78.03	72.35	75.27	75.35	78.49	**78.66**
	AUC	86.02	86.82	89.81	86.04	85.63	88.29	88.98	**90.53**
Slashdot	Micro-F1	82.65	83.89	85.90	84.72	82.96	84.94	**86.16**	85.03
	Binary-F1	89.18	89.83	91.05	90.70	89.26	90.55	**91.28**	90.59
	Macro-F1	72.73	75.54	**78.92**	73.99	74.03	76.71	**78.92**	77.63
	AUC	84.09	87.52	**90.17**	88.80	85.34	88.74	89.77	89.79
Epinions	Micro-F1	91.73	91.13	93.36	92.26	91.12	92.93	**93.55**	92.25
	Binary-F1	95.25	94.89	96.15	95.61	94.86	95.93	**96.28**	95.51
	Macro-F1	81.60	80.60	86.01	81.30	81.05	84.54	**86.10**	83.41
	AUC	88.72	90.95	93.51	94.17	87.45	93.33	94.11	**94.19**

Tables 5.2 and 5.3 report the results obtained for the existence and direction prediction tasks, respectively. The tables show that, when compared to the other 10 methods, SigMaNet achieves the best performance on 9 datasets out of 12 and that it achieves either the first- or the second-best performance on 12 datasets out of 12. SigMaNet is also consistently better than the state of the art on the synthetic datasets. This is likely due to the positive homogeneity property (Theorem 4) as the synthetic datasets have a significantly wider range of weights ($[2, 1000]$) than the real-world ones (in `Telegram`, for instance, the mean and median weight is 2 and 20.7 and only 96.4% of the weights are smaller than 100).

Table 5.2 Accuracy (%) on datasets of the existence prediction task

	Existence prediction					
	Telegram	Bitcoin Alpha*	Bitcoin-OTC*	$\alpha_{ij} = 0.05$	$\alpha_{ij} = 0.08$	$\alpha_{ij} = 0.1$
ChebNet	75.30±1.54	81.93±0.64	82.07±0.38	50.24±0.35	50.21±0.33	50.25±0.34
GCN	67.88±1.39	81.53±0.57	81.65±0.35	50.26±0.30	50.24±0.26	50.18±0.26
APPNP	68.52±5.76	81.62±0.57	81.02±0.51	60.62±0.46	62.61±0.64	63.51±1.93
SAGE	85.36±1.27	82.74±0.48	83.28±0.65	60.92±0.82	61.50±4.05	62.77±1.50
GIN	72.37±3.57	74.64±5.43	77.75±1.15	57.52±4.47	55.50±5.14	55.25±7.14
GAT	78.37±2.11	82.60±0.43	83.43±0.52	55.97±2.58	54.37±0.89	50.24±0.35
DGCN	82.97±2.06	83.13±0.61	83.79±0.36	55.41±3.09	55.70±5.71	56.15±5.65
DiGraph	82.15±1.11	83.24±0.38	**84.77±0.83**	59.09±3.66	57.64±2.35	58.66±3.28
DiGCL	78.80±1.50	80.22±0.77	81.99±0.62	60.69±0.27	60.63±0.18	60.49±0.15
MagNet	**86.32±1.06**	83.26±0.50	84.14±0.44	61.27±0.19	63.81±0.20	64.93±0.43
SigMaNet	84.95±0.95	**83.28±0.54**	84.71±0.39	**62.25±0.31**	**64.48±0.17**	**65.49±0.31**

Table 5.3 Accuracy (%) on datasets of the direction prediction task

	Direction prediction					
	Telegram	Bitcoin Alpha*	Bitcoin-OTC*	$\alpha_{ij} = 0.05$	$\alpha_{ij} = 0.08$	$\alpha_{ij} = 0.1$
ChebNet	78.56±3.53	53.86±1.15	50.06±1.04	50.13±0.30	50.23±0.25	50.13±0.30
GCN	63.86±1.40	55.32±1.12	49.63±1.82	50.05±0.15	50.24±0.29	50.13±0.30
APPNP	75.70±9.08	57.14±1.03	52.61±1.63	66.42±1.35	70.25±1.46	71.93±0.47
SAGE	91.15±0.77	55.82±1.60	55.29±1.23	66.62±1.72	68.84±2.38	69.43±6.79
GIN	80.77±5.01	56.04±1.42	53.31±1.58	60.51±6.88	60.87±9.50	57.66±9.04
GAT	84.06±11.17	55.20±1.06	53.23±0.63	52.71±1.53	57.07±1.50	57.43±1.07
DGCN	89.81±1.20	56.35±0.84	54.06±0.90	55.97±2.58	62.64±6.91	65.53±6.73
DiGraph	87.46±0.84	**58.62±1.09**	56.37±1.29	65.51±1.71	67.09±1.65	67.43±2.10
DiGCL	82.98±1.72	55.98±0.91	56.42±0.59	67.34±0.33	66.92±0.26	66.24±0.29
MagNet	**91.65±0.79**	56.84±0.74	55.63±0.74	68.50±0.23	72.01±0.33	73.28±0.37
SigMaNet	91.20±0.65	56.90±0.60	**57.19±0.58**	**69.10±0.18**	**72.74±0.23**	**73.77±0.18**

5.4.4 Node classification

The node classification task consists in predicting the class label to which each node belongs.

Also for this task, we only consider graphs with nonnegative edge weights, and thus compare SigMaNet not just to spatial GCNs, but also to the state-of-the-art spectral-based ones. Similarly to the previous two tasks, we will show that also for this task the wider applicability of SigMaNet does not hinder its performance.

We consider the `Telegram` dataset[4] as well as the three synthetic datasets. `Bitcoin-OTC` and `Bitcoin Alpha` dataset are not considered as they lack label information. We rely on the standard 60%/20%/20% split for training/validation/testing across all datasets. The experiments are run with k-cross validation, with $k = 10$.

Table 5.4 Testing accuracy (%) of node classification.

	Node classification			
	Telegram	$\alpha_{ij} = 0.05$	$\alpha_{ij} = 0.08$	$\alpha_{ij} = 0.1$
ChebNet	61.73±4.25	20.06±0.18	20.50±0.77	19.98±0.06
GCN	60.77±3.67	20.06±0.18	20.02±0.06	20.01±0.01
APPNP	55.19±6.26	33.46±7.43	34.72±14.98	36.16±14.92
SAGE	65.38±5.15	67.64±9.81	68.28±10.92	82.96±10.98
GIN	72.69±4.62	28.46±8.01	20.12±0.20	20.98±8.28
GAT	72.31±3.01	22.34±3.13	21.90±2.89	21.58±1.80
SSSNET	24.04±9.29	**91.04±3.60**	94.94±1.01	96.77±0.80
DGCN	71.15±6.32	30.02±6.57	30.22±11.94	28.40±8.62
DiGraph	71.16±5.57	53.84±14.28	38.50±12.20	34.78±9.94
DiGCL	64.62±4.50	19.51±1.21	20.24±0.84	19.98±0.45
MagNet	55.96±3.59	78.64±1.29	87.52±1.30	91.58±1.04
SigMaNet	**74.23±5.24**	87.44±0.99	**96.14±0.64**	**98.60±0.31**

The results are reported in Table 5.4. SigMaNet achieves notable performance on all four datasets, especially on the synthetic ones as the graph density increases, where being able to rely on both edge direction and weight information seems to be paramount for correct node labeling. This is confirmed by the extremely poor performance of ChebNet and GCN, which ignore the edge direction. When comparing SigMaNet to MagNet, SigMaNet achieves a consistently better performance of about 10% on average. This can be ascribed to the

[4]We point out that `Telegram` was also used for node classification in [224], but it is treated as an unweighted graph. In contrast, in our experiments the original topology of the graph is maintained.

positive homogeneity property of SigMaNet, which circumvents the sign-pattern inconsistency MagNet suffers from, which is likely to reduce its ability to adequately propagate the information between nodes. We note that, while SSSNET outperforms SigMaNet once by about 4%, SigMaNet outperforms SSSNET by about 50% on `Telegram`. SSSNET's poor performance on this dataset is likely due to the lack of seed nodes that SSSNET needs to identify and target node classes (which are present in the DSBM graphs).

5.5 Conclusion

We have extended the applicability of spectral GCNs to (directed) graphs with edges of weight unrestricted in sign by introducing the *Sign-Magnetic Laplacian* matrix. Thanks to its properties, which we rigorously derived, we embedded the matrix into a generalized convolution operator which is the cornerstone of our proposed spectral GCN: SigMaNet, which is the first spectral GCN capable of handling (directed) graphs with weights not restricted in sign nor magnitude. Compared with similar approaches presented in the literature, SigMaNet also does not suffer from any sign-pattern inconsistencies, making it capable to handle graphs with arbitrarily large weights (also in a completely parameter-free way and without preprocessing). Thanks to extensive numerical experiments, we have shown that, on graphs with negative weights where no other spectral GCN can be applied, SigMaNet's performance is either better or in line with more complex architectures, and that, on graphs with nonnegative weights where state-of-the-art spectral GCNs can be employed, SigMaNet's performance is consistently either the best or the second-best across all tasks.

Future Work. Extending the *Sign-Magnetic Laplacian* to multi-graphs and non-digon-free graphs without information loss (an issue shared by every Hermitian Laplacian matrix), and to hypergraphs, as well as experimenting with architectures featuring three or more convolutional layers.

6

On-Street Parking Prediction

The increasing number of cars in urban areas and the limited parking space have made it necessary to address this issue. For these reasons, this chapter focuses on analyzing the parking phenomenon in urban areas and predicting parking indicators. In particular, we created two quantitative indices - the *Average Parking Time* and the *Average Number of Vehicles Parked Simultaneously* - to describe parking from the time and quantity perspectives. We employed various Machine Learning and Deep Learning techniques, including statistical models, convolutional graph networks (GCNs), and convolutional neural networks (CNNs), to predict the values of these indicators. In our experiments, we found that the 3D-CLoST model excelled in accurately predicting parking indicators compared to other techniques. Interestingly, we also observed that statistical models were able to achieve performance levels that were very close to those of more complex models.

6.1 Introduction

As a result of the increase in motor vehicles, the limited availability of on-street parking and associated traffic congestion has become a major problem in urban transportation systems. Finding available parking, especially during peak hours, is a significant challenge faced by cities both big and small such as Paris, New York and even Santander [141]. The phenomenon known as "cruising for parking" [166] where drivers continuously search for a free parking spot, is commonly observed in urban areas with high parking demand. This not only causes delays for drivers, but also contributes to increased traffic congestion, as per an IBM survey, it's estimated that around 30% of urban road traffic consists of vehicles searching for parking spaces [83].

The issue of finding a parking spot not only causes inconvenience for drivers, but also harms the environment. Studies, such as [230], have shown that traffic congestion caused by searching for parking leads to increased pollution in urban areas. For example, in the city of Los Angeles, cars traveling at low speeds in search of a parking spot result in nearly 1.61 million vehicle miles traveled annually. This results in wasted time for

drivers, with an estimated 95,000 hours spent searching for parking, as well as significant fuel consumption, 47,000 gallons (or approximately 178,000 liters) leading to emissions of 730 tons of CO_2. A study conducted in Zurich [32], Switzerland, examines the impact of parking occupancy in the city. The work finds that in a small area of the city measuring 0.28 km^2 and experiencing a demand of 2,687 trips on a typical workday, vehicles searching for an available parking spot results in 83 additional hours of travel and 1,038 km of additional travel distance. The most challenging conditions are observed during the lunch hour, with the average time required to find a free parking spot being 13 minutes. As a result, the difficulty in finding parking amplifies problems including increased fuel consumption, pollution emissions, traffic congestion, and wasted time, but it also poses a danger to drivers, as their focus on finding parking can lead to accidents [103].

Prior knowledge of parking information can benefit both motorists and public administration. With this knowledge, the decision-maker can anticipate and address potential problems, and also suggest new parking areas to users based on accurate predictions. For this reason, a significant amount of research has been conducted in the area of parking management, with solutions such as the use of sensors to detect available parking spaces [140] and user feedback through apps to inform others of available parking spaces [149]. However, these solutions are limited as they rely on real-time data and do not enable the reservation and allocation of spaces, making them only useful for brief moments and when the user is near the parking area. Recently, in order to address this problem, predictive models have been developed to effectively predict parking-related indicators, such as the number of available free parking spaces at a specific time [165]. In addition, the increased connectivity of urban areas allows for the deployment of sensors to collect a large amount of data that can be used for analysis and decision-making [17]. This data can also be shared with drivers through mobile devices or integrated into vehicle infotainment systems [166] for real-time updates.

The introduction of predictive models has contributed to the emergence of two different categories: off-street and on-street parking prediction [9]. Off-street parking predictions are forecasting tasks performed with data regarding garages, surface lots, and other dedicated parking facilities. They are generally easier to model than the same on-street tasks because the information is more readily available and often known in advance. In contrast, tasks concerning on-street parking use data concerning public streets. Unlike off-street parking, the number of on-street parking spaces can vary depending on factors such as street width and the presence of sidewalks, and can be difficult to detect and measure. In these cases, machine learning models can be trained on data from various sources, such as street cameras, sensors, and GPS data.

As we have just analyzed, the topic of parking is very broad and encompasses a wide range of issues and factors that can affect parking systems. There are many aspects of this topic that can be studied in order to gain insights into the different elements that characterize this phenomenon. Unlike previous studies, this work investigates parking behavior in the city by analyzing and predicting two key indicators: the average parking time and the average number of vehicles parked simultaneously. These indexes provide valuable insights into the utilization and occupancy of parking lots in the city. For this reason, in particular, we focus on their prediction in the context of on-street parking, utilizing data collected from the GPS of vehicles. In order to make predictions, we used three different approaches commonly found in the literature: statistical models, CNN models, and GCN models. The purpose of this selection was to evaluate their differences in performance, as researches have shown that GCN models often outperform CNN and statistical models in the problem of flow prediction within the city [12, 169], but no comparison of this kind has been conducted before on these two indicators.

The rest of this Chapter is organized as follows. In Section 6.2 we define the on-street parking prediction problem and the proposed framework is described in detail. In Section 6.3, the literature on techniques used in flow and traffic prediction is analyzed. In Section 6.4 we describe the case study and in Section 6.5 data and

results of experiments are presented and analyzed. Finally, conclusions and recommendations for future work are discussed in Section 6.6.

6.2 Problem Statement and Methodology

Given a tessellation of the area of interest (i.e. the city) into regions of regular shape and a set of historical vehicle observations within the city, along with potential other spatial and non-spatial data sources for a reference time horizon T_H of H time points, the parking index prediction problem is defined as the problem of minimizing the prediction error for the selected parking index at the first time point after T_H, denoted as t'.

In this study, two indexes have been chosen to assess the parking phenomenon: *i)* the *Average parking time*, which reflects how long cars are parked in a particular area, and *ii)* the *Average number of vehicles parked simultaneously*, which can provide insight into the demand for parking lots in a particular area. These two indexes have a dynamic connotation, meaning that they can vary over time. They enable us to examine the parking phenomenon from two distinct perspectives: *i)* duration (average parking time) and *ii)* quantity (number of vehicles parked simultaneously).

Below is a comprehensive explanation of the two indices and the procedure for constructing the input matrices for the algorithms.

6.2.1 Index Definition

For this study, we consider two different indexes:

1. **Average parking time (S).** This index measures the mean time spent by parked vehicles in each area i and time slot t. It is defined as follows.

$$S_i^t = \frac{\sum_{j=0}^{V} s_{ij}^t}{V_i^t} \quad i \in \{1,...,I\} \quad \text{and} \quad t \in \{1,...,T\} \tag{6.1}$$

where s_{ij}^t is the parking time of vehicle j in area i at each time t, and V^t is the total number of vehicles that at some point in t found themselves parked in i. Figure 6.1 illustrates an example of the index.

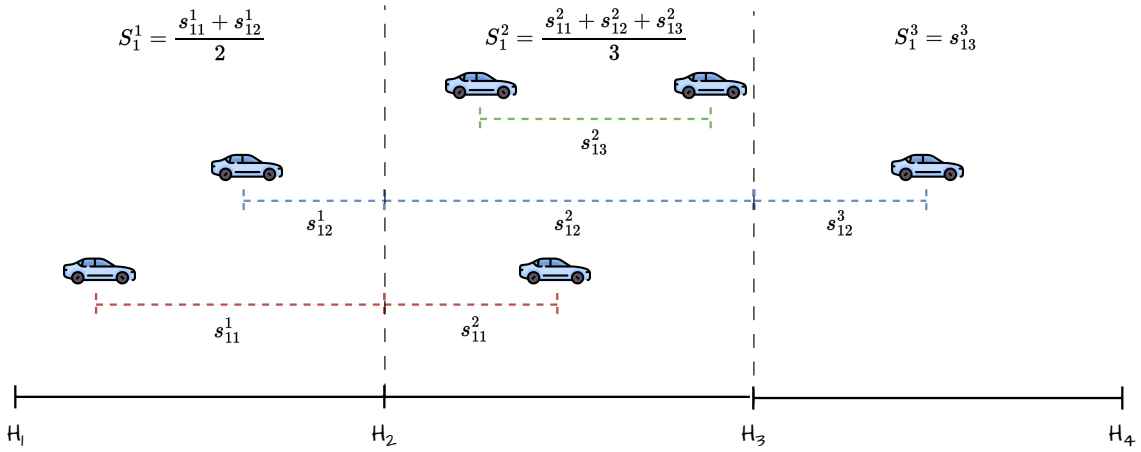


Figure 6.1 Example of the average parking time

2. **Average number of vehicles parked simultaneously (B).** This index measures the mean number of vehicles parked simultaneously in each area i and time slot t. It provides an indication of the simultaneously parked vehicles in different areas of the city. The equation for this index is shown below.

$$B_i^t = \frac{\sum_{k=0}^{K} f_i^{t^k}}{K^t} \quad \text{with} \quad i \in \{1,...,I\} \quad \text{and} \quad t \in \{1,...,T\} \tag{6.2}$$

where k is the time interval within t in which there are no changes in the number of parked vehicles, $f_i^{t^k}$ is the number of vehicles parked simultaneously in area i during k within t, and K^t is the total number of changes in parked vehicles number at each time slot t. An example of the index is shown in Figure 6.2.

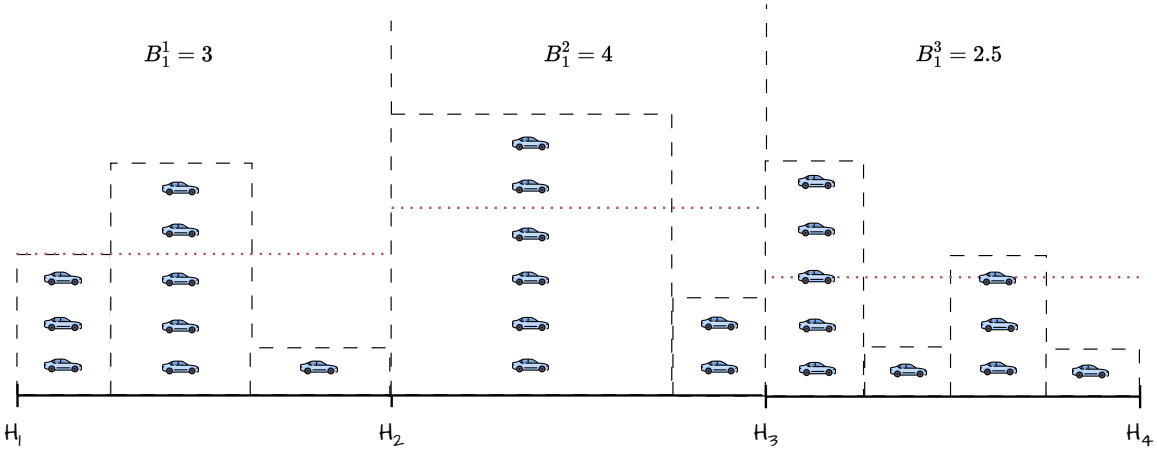


Figure 6.2 Example of the average number of vehicles parked simultaneously

6.2.2 Matrix Definition

The appropriate definition of input is a crucial step when using Deep Learning algorithms, as it forms the foundation of the model's ability to learn and make accurate predictions. In this specific case, we are utilizing both CNN and GCN models. These types of networks have different characteristics and require different types of input to function effectively. For this reason, we necessarily have to define two different matrices:

1. **Transition Matrix**
2. **Features Matrix**

Following, we will analyze in detail the steps for their creation.

Transition Matrix

The transition matrix (L) represents the probability of displacement between different areas within a city. It highlights the spatial correlation between different regions, remaining constant regardless of the index chosen, but varying in time. In the context of graph theory, it is known as the adjacency matrix. The size of this matrix is determined by the number of zones (nodes) in the city, so if there are N ($Z \times M$) regions, the matrix at time t is of size $L^t \in \mathbb{R}^{N \times N}$. Then, to take into account the temporal dependence, over the time horizon T_H (equally

divided into H time slots of duration ω), the transition representation is extended to a tensor of three dimensions $L \in \mathbb{R}^{H \times N \times N}$.

Denoting with τ_j the travel duration of vehicle j, each displacement is associated with a weight p_j, which can take on two different values:

$$p_j = \begin{cases} 1, & \text{if } \omega > \tau_j \\ \frac{\omega}{\tau_j}, & \text{otherwise.} \end{cases} \tag{6.3}$$

Therefore, trips that span across multiple time slots are given a lower weight, reflecting the fact that the displacement occurred over a longer period of time. As shown in Figure 6.3, vehicle three (*green line*) has a travel time shorter than the time slot duration, resulting in $p_3 = 1$. While, vehicles one (*red line*) and two (*blue line*) have travel times longer than ω, thus their weights are computed using the formula $\frac{\omega}{\tau_j}$.

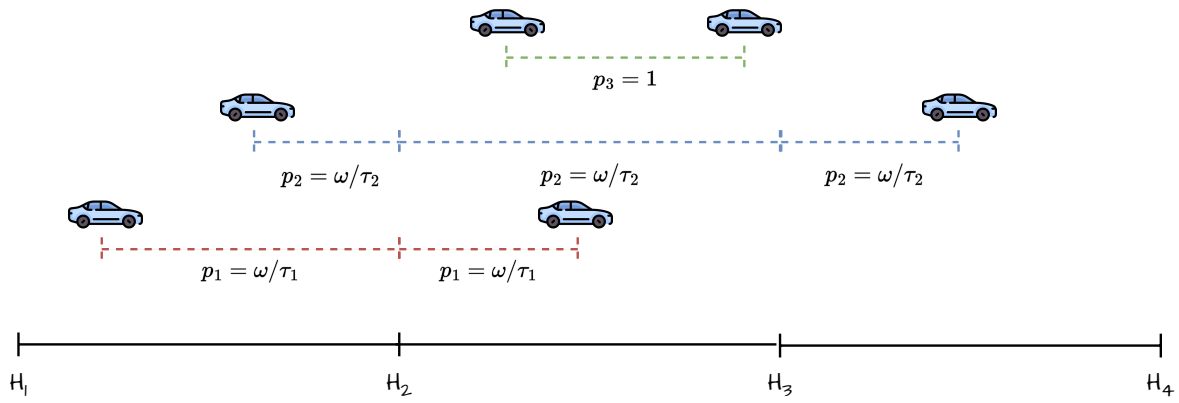


Figure 6.3 A schematic example of how weights are determined

The adjacency matrix, represented by the transition matrix, is a crucial component for graph models as it defines the rule for information propagation within the graph [134].

Features matrix

The values of the feature matrix (F), unlike the transition matrix, vary not only by time but also by the index used, as it has to contain the information that the models will use to make predictions. The construction of the feature matrix also varies depending on the model class. In particular:

- In graph-based models, the state of the index at time t can be represented by a matrix $F_t \in \mathbb{R}^{N \times C}$, where N is the total number of regions in the city and C indicates the variables considered in the analysis (in this specific case we consider separately the two indices, so $C = 1$). Then, to take into account the time horizon T_H (divided into H time points), we obtain a three-dimensional tensor $F \in \mathbb{R}^{H \times N \times C}$.
- In traditional convolutional models, the initial matrix F_t is transformed into a three-dimensional tensor $F_t^{'} \in \mathbb{R}^{Z \times M \times C}$, where $Z \times M$ is the total number of regions in the city and C is the number of features, in our case $C = 1$. Also in this case, to take into account the time horizon T_H, we create a four-dimensional tensor $F^{'} \in \mathbb{R}^{H \times Z \times M \times C}$.

6.3 Related Prior Work

The majority of the research and development were focused on addressing a specific problem: the recommendation of parking spots. One common solution is a recommendation system that uses real-time sensors to detect the availability of parking spaces and provides this information to users through a mobile phone application or other mean [140]. For instance, in [214] the authors evaluate a real-time Wireless Sensor Network (WSN) connected to a web server that collects data on available parking spots and sends it to users via a mobile app. Similarly, the Intelligent Parking Assistant (IPA) Intelligent Parking Assistant (IPA) is proposed in [13]. This system allows users to reserve a parking spot, but does not provide predictions on parking spot availability. Authors in [50] present a simulation-based method called Parking Rank that uses public information about parking spots, such as price and availability, to rank them using the Page Rank algorithm. However, these systems are limited in their ability to predict the availability of parking spaces in a specific area or time frame, as they rely on collecting data in real-time.

As a result, in recent years, several studies have proposed solutions using Machine Learning and Deep Learning techniques to solve problems that go beyond just recommending parking spots. In particular, related to the off-street prediction tasks, authors in [192] propose a Neural Network model to predict the occupancy rate of parking areas and parking spots. For instance, using this model, it is estimated that there is a 75% chance of finding a parking spot available within 5 minutes in a specific parking area. A study conducted in [9] compares different techniques such as Bayesian Regularized Neural Network, Support Vector Regression, Recurrent Neural Network, and Auto-regressive integrated moving average, for forecasting the availability of parking spots within a specific garage without identifying a specific spot. Their approach is limited to parking garages with gates (i.e., off-street parking spots) and also includes weather forecasts in their dataset. Authors in [227] compare Regression Tree, Neural Network, and Support Vector Regression (SVR) methods for predicting parking occupancy rates. They collected data that focused on information such as the number of occupied parking spaces. They found that the Regression Tree method was more effective than the other two methods they evaluated. An approach to predict the number of available parking spots utilizing recurrent neural networks (RNNs) is proposed in [31]. Authors improve the performance by introducing a Genetic Algorithm (GA) technique to search the best configuration of RNN. They used the parking data from Birmingham, U.K., which includes the parking occupancy rate for each parking area, given the time and date. Authors in [218] employee the Auto Regressive Integrated Moving Average (ARIMA) model to predict the number of available parking spaces. The ARIMA model is commonly used for making time series predictions. In their experiment, they utilized data from the underground parking of a central mall for one month, October 2010. However, using only one month's data may not provide a comprehensive understanding of the parking occupancy patterns as it could vary from month to month, and other factors such as public holidays can also impact the results. The study in [20] focuses on identifying occupied parking spots using image processing on video footage. This work uses real-time processing and does not provide any predictions for future occupancy. However, the approach can be useful for collecting data. Similarly, authors in [175] use computer vision techniques and camera sensors to detect parking spaces and identify their occupancy status. They carried out various steps including frame pre-processing, adaptive background subtraction, metrics and measurements, history creation, results merging for final classification, and parking space status. Like the previous work above, this one is not related to predicting the future availability of parking spots.

Unlike previous works, authors in [6] deal with the prediction of on-street parking in Santander, a Spanish smart city. Their approach focuses on forecasting the status of individual parking spaces, with a short-term prediction horizon of 10–20 minutes. This is based on the observation that during peak hours, the occupancy

status of parking spots near city centers or shopping malls tends to change frequently within 10-20 minute intervals, making a longer time horizon less accurate.

Our study also leverages on-street parking data, but unlike prior research that only predicts parking occupancy, our focus is on developing two indicators to reflect parking characteristics and conducting an experimental study comparing three diverse groups of predictive models.

6.4 Case Study

A real-world case study was considered for the experimental analysis: *RomaCar* dataset contains 15 million records of 28,513 vehicle pick-ups and drop-offs over a one-year period for the city of Rome, Italy. The vehicles in this dataset are equipped with On Board Unit (OBU) device, that records information related to vehicle position (latitude and longitude), time, GPS signal quality, distance from the last position, motor status (on, off, running), heading and instantaneous speed. The OBU device stores GPS measurements with an accuracy range of 10-30 meters every 2 kilometers traveled or, alternatively, every 30 seconds when the vehicle is on a freeway or major urban street. This dataset was collected by OctoTelematics in 2013 and the penetration rate of equipped vehicles in Rome is approximately 6% [116, 135].

The city of Rome has been divided into 2322 regular zones, with markers of 400 meters on each side, covering an area of approximately 450 km^2, extending from the city center to the outer ring road, as shown in Figure 6.4. The marker size used in this study is consistent with the ones used in other studies [2, 60, 206] and in our previous study conducted in the city of Milan, which was reported in Chapter 2.

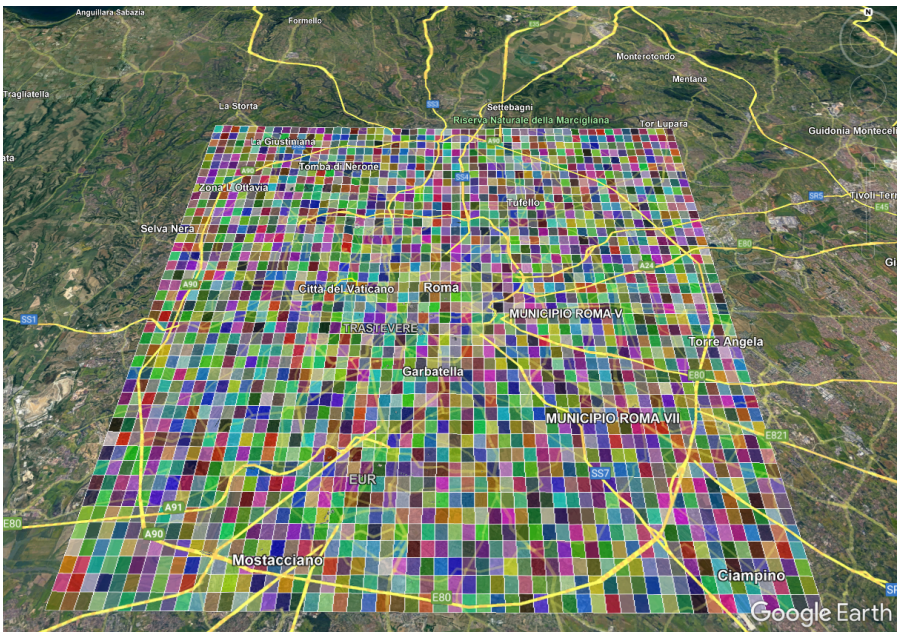


Figure 6.4 Map of Rome's tessellation

Each record in the dataset, representing a single displacement, is identified by the following parameters:

- *idtrajectory.* Unique number identifying the trip
- *idterm.* Unique number identifying the vehicle

- *from_zone_fid.* Unique number identifying the departure zone
- *from_timedate_gmt.* Date and time the trip began
- *to_zone_fid.* Unique number identifying the arrival zone
- *to_timedate_gmt.* Date and time the trip ended
- *tripdistance_m.* Distance traveled in meters
- *triptime_s.* Duration of the trip in seconds
- *stoptime_s.* Duration of the stop at the end of the trip in seconds.

6.4.1 Pre-processing

Before using the dataset for experiments, a pre-processing step was conducted. This involved removing all records that had a value less than 300 seconds for the column *stoptime_s*. This resulted in the removal of all stops shorter than 5 minutes from the original dataset. Additional columns were then added to enhance the available information. Specifically:

- *vel_km_h.* Calculated as the ratio of distance traveled to time spent, represents the average cruising speed.
- *end_stoptime.* Calculated by adding the *stoptime_s* column to the *to_timedate* column, represents the time when the stop ends.
- *in_out.* Column identifying whether the displacement occurred within the city or came from or is headed outside the analyzed areas. It takes the value 0 if the displacement is within the city, value 1 if the displacement is incoming, and -1 if the displacement is outgoing the city.

Finally, all records that have value equal to -1 in the *in_out* column are removed from the dataset. This choice is due to the fact that we are only interested in analyzing the value of indexes within the city of Rome. Therefore, at the end of the pre-processing phase, the dataset now consists of approximately 12 million records.

6.4.2 Descriptive Statics

Table 6.1 shows summary statistics for several variables related to parking and travel. These variables include "Parking Duration", "Speed", "Travel Time", "Trip Distance", and "Parked vehicles simultaneously". The statistics provided include the mean, median, upper quartile, lower quartile, and maximum value for each variable. The statistics show that on average, the vehicles in the dataset stay parked for around 7 hours. The distribution of parking duration is heavily skewed to the left, with 90% of parking durations being less than 15 hours. Since this analysis is focused on vehicles primarily traveling in urban areas, the average speed is 25 km/h with short travel times, around 18 minutes. As a result, the average distance traveled is also relatively short, at 8.63 kilometers. When analyzing the number of cars parked simultaneously, it appears that in most cases this number is not excessively high, on average there are about 8 vehicles parked simultaneously with the 90th percentile of the distribution being equal to 16. The unexpected result may be attributed to the fact that, as we have already discussed, our dataset represents approximately 6% of all cars in Rome. Overall, this data suggests that the vehicles in question primarily stay parked for short periods of time, travel relatively short distances, and tend to have a limited number of vehicles parked in the same location at the same time.

Table 6.1 Dataset statistics

Variable	Mean	Median	Upper Quartile	Lower Quartile	Max value
Parking Duration (h)	6.57	1.54	7.40	0.34	872
Speed (km/h)	25.46	21.01	30.93	14.68	143
Trip Time (min)	17.57	13.32	23.02	7.4	179
Trip Distance (kilometer)	8.63	4.62	10	2.1	199
Parked Vehicles simultaneously	8.50	8	12	4	59

6.5 Experimental Analysis

In this section, we discuss the results obtained from the experiments carried out to predict the two parking indexes: the average parking time and the average number of vehicles parked simultaneously.

6.5.1 Reference Methods

As previously stated, the experiments have been conducted by comparing three different types of approaches: statistical models, CNN models, and GCN models. For this experiment, most of the models evaluated were previously introduced in Chapter 4 for the flow prediction problem. Two additional statistical models and two GCN models were also included in the comparison. Specifically:

ARIMA [28]: this is a statistical model used for time series forecasting. It combines three components: *i)* autoregression (AR) component, which models the dependence between an observation and a number of lagged observations, *ii)* difference component (I, for "integrated"), which models the dependence between an observation and the differences between consecutive observations, and *iii)* moving average (MA) component, which models the dependence between an observation and a moving average of past errors or residuals.

Prophet [176]: it is a time series forecasting model developed by Facebook. It is based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It is designed to handle missing data, large outliers and make automatic adjustments for the effects of holidays and other events. The model is easy to interpret, and can handle time series with strong multiple seasonality and non-liner growth, which makes it a popular choice for many practitioners.

T-GCN [225]: this is a Deep Learning model that is designed to analyze temporal graph data. It combines the use of GCN and Gated Recurrent Unit (GRU) to learn representations of the data. GCN is used to extract features from the graph at each time step, while the GRU part is used to capture the temporal dependencies between the nodes.

T-SigMaNet: This model has a similar architecture to T-GCN, but with one key difference: instead of using a standard GCN to extract features from the graph, it uses our solution, SigMaNet [63]. As discussed in Chapter 5, SigMaNet is based on the *Sign-Magnetic Laplacian*, a positive semidefinite Hermitian matrix designed to handle (un)directed graphs with weights unrestricted in sign nor magnitude.

The models proposed for comparison were evaluated using two different metrics, *RMSE* and *MAPE*, which are defined as follows:

$$RMSE = \sqrt{\frac{\sum_{n=1}^{N} (\hat{y}_n - y_n)^2}{N}}$$

$$MAPE = 100 \cdot \frac{\sum_{n=1}^{N} \left| \frac{(\hat{y}_n - y_n)}{y_n} \right|}{N}$$

where $\hat{y}_n$ is the predicted parking index for region n at time slot t' and N is the total number of regions in the city.

Each experiment was repeated ten times (replicas) with a different random seed in each replica in order to reduce the natural stochasticity of learning-based models. Mean and standard deviation are reported for each metric to give a reliable indication of the general performance of the compared methods. In order to make a fair comparison, it was necessary to search for the optimal configuration of hyperparameters for all approaches considered through the *Bayesian optimization* technique [167]. Finally, in the experiment conducted with the graph-based models, only one adjacency matrix has been used, computed as the average of all L^t matrices.

Finally, throughout the tables contained in this section, the best results are reported in **boldface** and the second best are underlined.

6.5.2 Average Parking Time

The optimized parameters and the relative values used in the training phase are briefly summarized below for each model.

- **ST-ResNet.** Optimized parameters: number of residual units, batch size and learning rate. Optimal values found: 2, 8 and 0.001.
- **MST3D.** Optimized parameters: batch size and learning rate. Optimal values found: 16 and 0.00014.
- **PredCNN.** Optimized parameters: encoder length, decoder length, number of hidden units, batch size and learning rate. Optimal values found: 3, 2, 64, 16 and 0.00062.
- **ST-3DNet.** Optimized parameters: number of residual units, batch size and learning rate. Best values found: 4, 32 and 0.0007.
- **STAR.** Optimized parameters: number of remaining units, batch size and learning rate. Optimal values found: 6, 8 and 0.001.
- **3D-CLoST.** Optimized parameters: number of LSTM layers, number of hidden units in each LSTM layer, batch size, and learning rate. Optimal values found: 2, 500, 16, and 0.001.
- **STREED-Net.** Optimized parameters: kernel size, batch size, and learning rate. Optimal values found: 4, 16 and 0.00067.
- **T-GCN** Optimized parameters: batch size, number of filters, and learning rate. Optimal values found: 8, 64, and 0.01.
- **T-DGCN** Optimized parameters: kernel size, batch size, and learning rate. Optimal values found: 3, 64 and 0.00086.

Table 6.2 Results obtained for the Average Parking Time

Model	RMSE	MAPE
HA	14.92	24.01
ARIMA	8.99	19.39
Prophet	13.53	22.92
ST-ResNet	8.14 ± 0.009	17.53 ± 0.032
MST3D	13.23 ± 0.05	21.19 ± 0.095
3D-CLoST	$\mathbf{7.14 \pm 0.38}$	$\mathbf{9.99 \pm 0.85}$
PredCNN	8.32 ± 0.007	18.04 ± 0.105
ST-3DNet	8.14 ± 0.04	17.75 ± 0.034
STAR	8.02 ± 0.011	17.46 ± 0.065
STREED-Net	8.29 ± 0.018	17.53 ± 0.035
T-GCN	7.86 ± 0.011	17.35 ± 0.042
T-SigMaNet	8.12 ± 0.006	17.87 ± 0.015

Table 6.2 reports the results of the average parking time experiment. The best performance is achieved by the 3D-CLoST model in both metrics, while the HA model has the worst one. The second-best model is T-GCN, which is 10% and 73% behind 3D-CLoST in RMSE and MAPE, respectively. The best model in this experiment is based on Convolutional Neural Networks, specifically, it was the only method that combined both 3D convolutions and LSTMs. This architecture effectively extracts spatial and temporal information from the dataset, thanks to the capability of 3D convolutions to capture both spatial and temporal information, and LSTM's ability to process sequential information. Analyzing the performances of the three statistical models, it is interesting to see that the simplest one (HA) performs worse than the two more advanced methods (ARIMA and Prophet). Based on only time series, the three statistical models perform similarly to Deep Learning models that also take into account spatial relationships between areas. In order to understand this result, two different aspects must be considered. First, the Coefficient of Variation (CV) [127] of the index, calculated as $CV = \frac{Standard\ Deviation}{Mean}$, is 0.37, indicating low variability. Thus, it is relatively easy for the statistical models to identify successive values. Second, unlike the displacement prediction problem [223], the analyzed index does not appear to have a strong spatial correlation. This last observation is supported by the fact that the states of the vehicular flow in nearby regions are strongly correlated with each other, whereas, the parking index does not feature such a spatial dependency.

6.5.3 Average number of vehicles parked simultaneously

As before, the optimized parameters and the relative values used in the training phase for each model are briefly summarized below.

- **ST-ResNet.** Optimized parameters: number of residual units, batch size and learning rate. Optimal values found: 4, 8 and 0.00049.
- **MST3D.** Optimized parameters: batch size and learning rate. Optimal values found: 16 and 0.0003.
- **PredCNN.** Optimized parameters: encoder length, decoder length, number of hidden units, batch size and learning rate. Optimal values found: 3, 2, 32, 16 and 0.00047.
- **ST-3DNet.** Optimized parameters: number of residual units, batch size and learning rate. Best values found: 6, 16 and 0.0004.

- **STAR.** Optimized parameters: number of remaining units, batch size and learning rate. Optimal values found: 3, 8 and 0.0007.
- **3D-CLoST.** Optimized parameters: number of LSTM layers, number of hidden units in each LSTM layer, batch size, and learning rate. Optimal values found: 2, 150, 16, and 0.0003.
- **STREED-Net.** Optimized parameters: kernel size, batch size, and learning rate. Optimal values found: 4, 16 and 0.00086.
- **T-GCN.** Optimized parameters: batch size, number of filters, and learning rate. Optimal values found: 8, 64, and 0.0637.
- **T-SigMaNet.** Optimized parameters: kernel size, batch size, and learning rate. Optimal values found: 3, 64 and 0.00086.

Table 6.3 Results obtained for the Average number of vehicles parked simultaneously

Model	RMSE	MAPE
HA	2.57	48.04
ARIMA	2.43	37.84
Prophet	2.44	41.82
ST-ResNet	2.06 ± 0.015	36.12 ± 1.64
MST3D	2.32 ± 0.013	40.08 ± 0.51
3D-CLoST	$\mathbf{1.10 \pm 0.002}$	$\mathbf{31.88 \pm 0.01}$
PredCNN	2.49 ± 0.004	42.45 ± 0.89
ST-3DNet	2.01 ± 0.002	33.86 ± 0.33
STAR	2.02 ± 0.005	34.57 ± 0.72
STREED-Net	2.04 ± 0.01	35.09 ± 1.14
T-GCN	2.09 ± 0.03	38.53 ± 0.09
T-SigMaNet	2.08 ± 0.001	40.01 ± 0.03

Table 6.3 reports the results of the Average number of vehicles parked simultaneously. Also in this experiment, the best performance was achieved by the 3D-CLoST model in both metrics, while, in this case, the HA model has the worst one. The second-best model, ST-3DNet, performed 82% and 6% worse than 3D-CLoST in terms of RMSE and MAPE, respectively. The performance of the three statistical models is comparable to Deep Learning models. This result is in line with the previous experimentation and confirms that spatial relationships play a minor role in the index prediction problem, allowing statistical methods to perform similarly to Deep Learning models. Furthermore, the coefficient of variation for this index is 0.87, indicating a relatively stable level of fluctuation. This reduced variability makes it easy for statistical models to accurately identify and predict the values of the average number of vehicles parked simultaneously in different areas.

6.5.4 Threats to Validity

In this study, we examined parking patterns in Rome by analyzing and predicting the average parking time and the number of vehicles parked simultaneously. Our findings indicate that simple statistical models can perform as well as more complex ones. This result can be attributed to two factors regarding the indices: *i)* they have low variability, and *ii)* they do not show a strong spatial relationship. However, the conclusions drawn should not be considered general. Some important issues can be pointed out:

- The experimentation was done by considering only two indices, it is necessary to increase the number of indices to evaluate if the conclusions about low variability and lower impact of spatial relationships can be generalized to the parking phenomenon.
- The dataset used represents only a very small percentage, around 6%, of the vehicles circulating in the city of Rome.
- The best model is based on 3D convolution and LSTM, but no experimentation was conducted to determine which component is mainly responsible for the performance obtained.

Furthermore, it is worth noting that the 12 models proposed in the comparison were not specifically designed for this type of task, but all models except the statistical models were originally intended for traffic prediction. Although both tasks are spatio-temporal problems, the displacement prediction problem, as we have already analyzed above, has a strong spatial continuity component that is missing in parking indices.

Finally, the experimental campaign could be extended to other state-of-the-art architectures and techniques, such as transformer-based models.

6.6 Conclusion

The field of parking is becoming increasingly important within the Smart Mobility landscape, as it is a research area with significant social impact. However, it poses many challenges due to its complexity and lack of exploration, especially regarding on-street parking prediction. In this chapter, we first examined two indices (average parking time, and average number of vehicles parked simultaneously) that provide valuable insights into the utilization and occupancy of parking lots in the city. Then, through an experimental campaign, we compared three different families of prediction approaches (statistical models, CNN models and GCN models) for predicting these two indices to evaluate the models' performance in this specific context. The results indicate that 3D-CLoST consistently outperforms other models in predicting both indexes, while the performance of statistical models is on par with the other two types of models. This is a direct result of the high significance of the temporal component, as opposed to the spatial component, shown by the analyzed indices.

Future Work. Extending the experimental campaign by testing the model on other indices and datasets from different cities. This would provide a broader understanding of the model's performance and its ability to generalize to different urban environments. Furthermore, it would be valuable to explore other cutting-edge Neural Network architectures and techniques, such as transformer-based models, to determine their potential for enhancing performance in this research field.

7

Conclusion and Outlook

Smart mobility solutions include a wide range of technologies and strategies, such as autonomous vehicles, intelligent transportation systems, and shared mobility alternatives. These technologies have the potential to greatly improve the efficiency and safety of our transportation systems, by reducing congestion and accidents, and making it easier for people to move. Moreover, Smart Mobility can reduce emissions and improve air quality as well as provide greater accessibility for all members of society, including those with disabilities or limited mobility. The benefits of Smart Mobility make it a worthwhile investment for cities and governments. In light of the ongoing development of advancing technologies and the growing demand for more efficient and sustainable transportation, it has the potential to play a crucial role in shaping the future of transportation. As a result, Smart Mobility is a valuable solution to improving the way people move, and it contributes to a more sustainable, livable, and efficient future.

In particular, this work focuses on bridging Artificial Intelligence and Transportation Science by delivering theoretical contributions as well as novel technical solutions to mobility problems. We demonstrated how the proposed solutions outperform the literature and try to answer some of the currently open research questions. In Chapters 2 and 3, emphasis is initially placed on data processing and explaining mobility data. In Chapters 4, 5, and 6, the focus shift towards the implementation and application of advanced Deep Learning techniques capable of better capturing spatial and temporal dependencies.

In Chapter 1, we identified several key research questions that shaped the direction of this thesis. In order to provide a comprehensive overview of our findings and demonstrate their relevance, we provide here specific answers to each question.

Research Question 1. *Can we predict the level of adoption of shared vehicles in a city, by analyzing and identifying the factors that drive their usage?*

In Chapter 2 and in Fiorini et al. [61], we demonstrate that the adoption of shared vehicles, specifically electric mopeds, can be predicted by considering certain factors, other than those related directly to users. Indeed,

our findings suggest that the use of e-mopeds is also driven by elements related to the built environment and demographic aspects of each neighborhood. In details, we considered four key features: three concerning the geographic characteristics (distance from center, walkability, concentration of places) and one about the population (education index). The results obtained on a real-world case study in the urban city of Milan reveal the strong impact these factors have in determining the adoption of e-moped sharing services.

Research Question 2: *Can we create an algorithm that can efficiently identify urban communities, by considering a set of factors?*

Characterizing urban communities is essential for understanding citizens' needs and neighborhood-wise dynamics. In Chapter 3 and in Fiorini et al. [60], we provide a multi-objective optimization algorithm that is able to identify behavioral communities by considering several factors, including population mobility patterns, neighborhood structural characteristics (via Map Embeddings), and distance between areas. The efficacy of this approach is validated through a practical application of the algorithm on a real-world dataset. The results showcase the effectiveness of the algorithm in defining meaningful communities.

Research Question 3: *Can we develop and efficiently implement novel Deep Learning architectures that enhance the ability to extract information from spatio-temporal mobility data?*

Predicting the number of incoming and outgoing vehicles for different city areas is challenging due to the nonlinear spatial and temporal dependencies typical of urban mobility patterns. In Chapter 4 and in Fiorini et al. [59, 62], we propose two different Deep Learning methods: *i)* 3D-CLoST, a model that exploits the synergy between 3D convolution and LSTM networks, and *ii)* STREED-Net, a novel autoencoder architecture featuring time-distributed convolutions, cascade hierarchical units and two distinct attention mechanisms (one spatial and one temporal). The idea behind these two architectures is to find a way to effectively captures and exploits complex spatial and temporal patterns in mobility data for the short-term flow prediction problem. The results of an extensive experimental analysis, conducted on three real-world datasets, indicate that one of our proposals, STREED-Net, improves the state of the art for this specific task.

Research Question 4: *Can spectral Graph Convolutional Networks solve mobility problems that involve the management of directed graphs with both positive and negative weights of arbitrary magnitude?*

In Chapter 5 and in Fiorini et al. [63], we introduce SigMaNet, a generalized Graph Convolutional Network capable of handling both undirected and directed graphs with weights not restricted in sign nor magnitude. The cornerstone of SigMaNet is the *Sign-Magnetic Laplacian* (L^{σ}), a new Laplacian matrix that we introduce *ex novo* in this work. L^{σ} allows us to bridge a gap in the current literature by extending the theory of spectral GCNs to (directed) graphs with both positive and negative weights. L^{σ} exhibits several desirable properties, including the encoding of the edge direction and weight in a clear and natural way that is not negatively affected by the weight magnitude. These properties are not enjoyed by other Laplacian matrices, on which several state-of-the-art architectures are based. L^{σ} is also completely parameter-free, which is not the case of other Laplacian operators such as, e.g., the *Magnetic Laplacian*. Our proposed approach's versatility and performance are amply demonstrated via computational experiments. Indeed, our results show that, for at least a metric, SigMaNet achieves the best performance in 15 out of 21 cases and either the first- or second-best performance in 21 cases out of 21, even when compared to architectures that are either more complex or that, due to being

designed for a narrower class of graphs, should—but do not—achieve better performance.

Research Question 5: *Can Deep Neural Networks and Statistical Models accurately predict the indices that describe the pattern of city parking?*

As a result of the increase in motor vehicles, the limited availability of on-street parking and associated traffic congestion has become a major problem in urban transportation system. There are several elements that characterize the phenomenon of parking that can be studied. In Chapter 6, we compared three different categories of prediction models, namely statistical models, CNN models, and GCN models, to address the parking index problem of predicting the value of a specific parking indicator. Specifically, two indices that are able to evaluate the parking phenomenon were used: average parking time and average number of vehicles parked simultaneously. The results showed minimal performance variation among the different model categories, even if our proposed model, 3D-CLoST, consistently outperformed the others. This limited diversity in results can be attributed to two factors related to the indices: *i)* low variability, and *ii)* weak spatial relationship.

Our examination of Smart Mobility has emphasized the critical role of AI in addressing mobility challenges: AI offers the potential to resolve persistent problems in the field. However, we have also observed that some issues, like displacement prediction, receive more attention, while others are neglected due to a lack of data or limited interest. To ensure that AI solutions for Smart Mobility are effective, it is important to take a more comprehensive approach that seeks a trade-off between algorithm performance and other critical factors such as data quality, explainability, and generalizability. Furthermore, the development of AI-based solutions for Smart Mobility requires close collaboration between researchers, industry experts, and policymakers. The integration of AI into the field of mobility requires a thorough understanding of the challenges and opportunities, as well as the technical, regulatory, and social implications. Collaboration is crucial to develop and apply AI to achieve the goals of Smart Mobility, such as increased efficiency, accessibility, safety, and sustainability.

7.1 Outlook

Despite the significant progress made in this field, there are still a number of challenges and opportunities for further research. Below are reported some of the main challenges that need to be addressed in the field of Smart Mobility. Many time-sensitive applications such as connected vehicles, smartphone applications, and many more in a smart city context require real-time or near-real-time data analytics. New analytic frameworks that allow advanced data analytics, as well as streaming data analytics, are required for these applications. Therefore, designing an effective data-driven approach to adapt the time-series model [158] for next-generation mobile, IoT, or resource-constrained devices and applications could be another major issue in the area. Implementing efficient solutions for dynamic routing in transportation [174] is crucial for the success of Smart Mobility systems. Several key steps are involved in this process, such as estimating user travel demands and optimizing resources. To ensure citizens have accurate, up-to-date information on the best routes and modes of transportation, considering factors such as traffic, weather, and disruptions, real-time routing, and scheduling is essential. In certain circumstances, the typical Machine Learning techniques may not be effective in building an analytical city model. It depends on data characteristics, problem nature, as well as target solution. In a rule-based system, for example, the association rule learning technique [1] extracts redundant generation from the data, making the decision-making process complicated and unproductive [157]. Therefore, a deeper understanding of the strengths and limitations of existing learning methods is required, and proposing

new techniques as well as their ensembles could be a promising direction for data-driven Smart City research. Lastly, the first/last mile problem [177], which is the lack of connectivity between public transportation and an individual's starting/destination point. This disconnect can greatly impact the overall convenience and efficiency of public transportation, making it difficult for individuals to fully utilize the available services. To address this, efficient smart mobility solutions must provide door-to-door connectivity, regardless of the mode of transportation.

Bibliography

[1] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499. Santiago, Chile, 1994.

[2] Álvaro Aguilera-García, Juan Gomez, and Natalia Sobrino. Exploring the adoption of moped scooter-sharing systems in spanish urban areas. *Cities*, 96:102424, 2020.

[3] Álvaro Aguilera-García, Juan Gomez, Natalia Sobrino, and Juan José Vinagre Díaz. Moped scooter sharing: Citizens' perceptions, users' behavior, and implications for urban mobility. *Sustainability*, 13 (12):6886, 2021.

[4] Muhammad Tayyab Asif, Justin Dauwels, Chong Yang Goh, Ali Oran, Esmail Fathi, Muye Xu, Menoth Mohan Dhanya, Nikola Mitrovic, and Patrick Jaillet. Spatiotemporal patterns in large-scale traffic speed prediction. *IEEE Transactions on Intelligent Transportation Systems*, 15(2):794–804, 2013.

[5] Austin Texas Open Data. Shared Micromobility Vehicle Trips. Accessed March. 27, 2023 [Online]. URL https://data.austintexas.gov/Transportation-and-Mobility/Shared-Micromobility-Vehicle-Trips/7d8e-dm7r.

[6] Faraz Malik Awan, Yasir Saleem, Roberto Minerva, and Noel Crespi. A comparative analysis of machine/deep learning models for parking space availability prediction. *Sensors*, 20(1):322, 2020.

[7] Abdelhadi Azzouni and Guy Pujolle. A long short-term memory recurrent neural network framework for network traffic matrix prediction. *arXiv preprint arXiv:1705.05690*, 2017.

[8] Lars Backstrom and Jure Leskovec. Supervised random walks: predicting and recommending links in social networks. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 635–644, 2011.

[9] Claudio Badii, Paolo Nesi, and Irene Paoli. Predicting available parking slots on critical and regular services by exploiting a range of open data. *IEEE Access*, 6:44059–44071, 2018. doi: 10.1109/ACCESS.2018.2864157.

[10] Kwangho Baek, Hyukseong Lee, Jin-Hyuk Chung, and Jinhee Kim. Electric scooter sharing: How do people value it as a last-mile transportation mode? *Transportation Research Part D: Transport and Environment*, 90:102642, 2021.

[11] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[12] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in neural information processing systems*, 33:17804–17815, 2020.

[13] Rosamaria Elisa Barone, Tullio Giuffrè, Sabato Marco Siniscalchi, Maria Antonietta Morgano, and Giovanni Tesoriere. Architecture for parking management in smart cities. *IET Intelligent Transport Systems*, 8(5):445–452, 2014.

[14] J. Robert Barth. *Chapter V. Symbol and Romanticism*, pages 105–127. Princeton University Press, 2015.

[15] Sandy Baum, Jennifer Ma, and Kathleen Payea. Education pays, 2013: The benefits of higher education for individuals and society. trends in higher education series. *College Board*, 2013.

[16] Henrik Becker, Francesco Ciari, and Kay W Axhausen. Comparing car-sharing schemes in switzerland: User groups and usage patterns. *Transportation Research Part A: Policy and Practice*, 97:17–29, 2017.

[17] Clara Benevolo, Renata Paola Dameri, and Beatrice D'Auria. Smart mobility in smart city. In Teresina Torre, Alessio Maria Braccini, and Riccardo Spinelli, editors, *Empowering Organizations*, pages 13–28, Cham, 2016. Springer International Publishing. ISBN 978-3-319-23784-8.

[18] Clara Benevolo, Renata Paola Dameri, and Beatrice D'auria. Smart mobility in smart city. In *Empowering organizations*, pages 13–28. Springer, 2016.

[19] Simone Bianco, Remi Cadene, Luigi Celona, and Paolo Napoletano. Benchmark analysis of representative deep neural network architectures. *IEEE Access*, 6:64270–64277, 2018.

[20] Nazia Bibi, Muhammad Nadeem Majid, Hassan Dawood, and Ping Guo. Automatic parking space detection system. In *2017 2nd international conference on multimedia and image processing (ICMIP)*, pages 11–15. IEEE, 2017.

[21] Tomasz Bieliński and Agnieszka Ważna. Electric scooter sharing and bike sharing user behaviour and characteristics. *Sustainability*, 12(22):9640, 2020.

[22] J Mark Bishop. Artificial intelligence is stupid and causal reasoning will not fix it. *Frontiers in Psychology*, 11:513474, 2021.

[23] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10): P10008, 2008.

[24] Geoff Boeing. Urban spatial order: Street network orientation, configuration, and entropy. *Applied Network Science*, 4(1):1–19, 2019.

[25] Boston Consulting Group. The road to autonomous vehicles must be paved with collaboration among all stakeholders. Accessed March. 27, 2023 [Online]. URL https://www.globenewswire.com/news-release/2015/09/08/924237/0/en/The-Road-to-Autonomous-Vehicles-Must-Be-Paved-With-Collaboration-Among-All-Stakeholders.html.

[26] Alexandre Bovet and Peter Grindrod. The activity of the far right on telegram, 2020.

[27] Howard Bowen. *Investment in learning: The individual and social value of American higher education*. Routledge, 2018.

[28] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[29] Jason Brownlee. A gentle introduction to pooling layers for convolutional neural networks. *Machine Learning Mastery*, 22, 2019.

[30] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018.

[31] Andrés Camero, Jamal Toutouh, Daniel H Stolfi, and Enrique Alba. Evolutionary deep learning for car park occupancy prediction in smart cities. In *International conference on learning and intelligent optimization*, pages 386–401. Springer, 2019.

[32] Jin Cao and Rashid Waraich. Impacts of the urban parking system on cruising traffic and policy development: the case of zurich downtown area, switzerland. *Transportation*, 2017. doi: 10.1007/s11116-017-9832-9.

[33] Annie Chang, Luis Miranda-Moreno, Lijun Sun, and Regina Clewlow. Trend or Fad? Deciphering the Enablers of Micromobility in the U.S. *A Report of SAE International*, 2019.

[34] Cen Chen, Kenli Li, Sin G Teo, Guizi Chen, Xiaofeng Zou, Xulei Yang, Ramaseshan C Vijay, Jiashi Feng, and Zeng Zeng. Exploiting spatio-temporal correlations with multiple 3d convolutional neural networks for citywide vehicle flow prediction. In *2018 IEEE international conference on data mining (ICDM)*, pages 893–898. IEEE, 2018.

[35] Siheng Chen, Rohan Varma, Aliaksei Sandryhaila, and Jelena Kovačević. Discrete signal processing on graphs: Sampling theory. *IEEE transactions on signal processing*, 63(24):6510–6523, 2015.

[36] Yiqi Chen, Tieyun Qian, Huan Liu, and Ke Sun. "Bridge": Enhanced Signed Directed Network Embedding. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 773–782, 2018.

[37] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[38] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. American Mathematical Soc., 1997.

[39] Alessandro Ciociola, Michele Cocca, Danilo Giordano, Luca Vassio, and Marco Mellia. E-scooter sharing: Leveraging open data for system design. In *2020 IEEE/ACM 24th International Symposium on DS-RT*, pages 1–8, 2020. doi: 10.1109/DS-RT50469.2020.9213514.

[40] Annalisa Cocchia. Smart and digital city: A systematic literature review. *Smart city*, pages 13–43, 2014.

[41] Marios Constantinides, Sagar Joglekar, Sanja Šćepanović, and Daniele Quercia. Imagine a walkable city: Physical activity and urban imageability across 19 major cities. *EPJ Data Science*, 10(1):56, 2021.

[42] Antoine Coutrot, E Manley, S Goodroe, C Gahnstrom, G Filomena, D Yesiltepe, R Dalton, J Wiener, C Hölscher, M Hornberger, and H Spiers. Entropy of city street networks linked to future spatial navigation ability. *Nature*, March 2022.

[43] Cyrille Médard de Chardon, Geoffrey Caruso, and Isabelle Thomas. Bicycle sharing system 'success' determinants. *Transportation research part A: policy and practice*, 100:202–214, 2017.

[44] Stefano de Luca and Roberta Di Pace. Modelling users' behaviour in inter-urban carsharing program: A stated preference approach. *Transportation Research Part A: Policy and Practice*, 71:59–76, 2015. ISSN 0965-8564. doi: https://doi.org/10.1016/j.tra.2014.11.001. URL https://www.sciencedirect.com/science/article/pii/S0965856414002675.

[45] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[46] Jutta Degele, Anna Gorr, Katja Haas, Dimitri Kormann, Sascha Krauss, Paulina Lipinski, Muhammet Tenbih, Christine Koppenhoefer, Jan Fauser, and Dieter Hertweck. Identifying e-scooter sharing customer segments using clustering. In *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pages 1–8. IEEE, 2018.

[47] Tyler Derr, Yao Ma, and Jiliang Tang. Signed graph convolutional networks. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 929–934. IEEE, 2018.

[48] Vasant Dhar. Data science and prediction. *Commun. ACM*, 56(12):64–73, dec 2013. ISSN 0001-0782. doi: 10.1145/2500499. URL https://doi.org/10.1145/2500499.

[49] Iain Docherty, Greg Marsden, and Jillian Anable. The governance of smart mobility. *Transportation Research Part A: Policy and Practice*, 115:114–125, 2018.

[50] Shi Dong, Mingsong Chen, Lei Peng, and Huiyun Li. Parking rank: A novel method of parking lots sorting and recommendation based on public information. In *2018 IEEE International Conference on Industrial Technology (ICIT)*, pages 1381–1386. IEEE, 2018.

[51] Norman R Draper and Harry Smith. *Applied regression analysis*, volume 326. John Wiley & Sons, New York, 1998.

[52] Ezgi Eren and Volkan Emre Uz. A review on bike-sharing: The factors affecting bike-sharing demand. *Sustainable Cities and Society*, 54:101882, 2020.

[53] European Commission. BIG IoT - Bridging the Interoperability Gap of the Internet of Things. Accessed March. 27, 2023 [Online]. URL https://cordis.europa.eu/project/id/688038.

[54] Michaël Fanuel, Carlos M Alaíz, Ángela Fernández, and Johan AK Suykens. Magnetic eigenmaps for the visualization of directed networks. *Applied and Computational Harmonic Analysis*, 44(1):189–199, 2018.

[55] Michaël Fanuel, Carlos M. Alaíz, and Johan A. K. Suykens. Magnetic eigenmaps for community detection in directed networks. *Physical Review E*, 95(2), Feb 2017. ISSN 2470-0053.

[56] Stefano Fiorini. SigMaNet: One Laplacian to Rule Them All, 2023. URL https://github.com/stefa1994/sigmanet.

[57] Stefano Fiorini and Marco Fagioli. Community Identification, 2021. URL https://github.com/UNIMIBInside/Community-Identification.

[58] Stefano Fiorini and Roberto Tagliabue. Smart Mobility Prediction, 2022. URL https://github.com/UNIMIBInside/Smart-Mobility-Prediction/.

[59] Stefano Fiorini, Giorgio Pilotti, Michele Ciavotta, and Andrea Maurino. 3d-clost: A cnn-lstm approach for mobility dynamics prediction in smart cities. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 3180–3189. IEEE, 2020.

[60] Stefano Fiorini, Michele Ciavotta, and Andrea Maurino. A multi-criteria algorithm for automatic detection of city communities. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 1266–1271. IEEE, 2021.

[61] Stefano Fiorini, Michele Ciavotta, Sagar Joglekar, Sanja Šćepanović, and Daniele Quercia. On the adoption of e-moped sharing systems. *EPJ Data Science*, 11(1):46, 2022.

[62] Stefano Fiorini, Michele Ciavotta, and Andrea Maurino. Listening to the city, attentively: A spatio-temporal attention-boosted autoencoder for the short-term flow prediction problem. *Algorithms*, 15(10), 2022. ISSN 1999-4893.

[63] Stefano Fiorini, Stefano Coniglio, Michele Ciavotta, and Enza Messina. Sigmanet: One laplacian to rule them all. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence*, 2023. (in press).

[64] Mauro Francini, Lucia Chieffallo, Annunziata Palermo, and Maria Francesca Viapiana. Systematic literature review on smart mobility: A framework for future "quantitative" developments. *Journal of Planning Literature*, 36(3):283–296, 2021.

[65] Lewis M. Fulton. Three revolutions in urban passenger travel. *Joule*, 2(4):575–578, 2018. ISSN 2542-4351.

[66] Satoshi Furutani, Toshiki Shibahara, Mitsuaki Akiyama, Kunio Hato, and Masaki Aida. Graph signal processing for directed graphs based on the hermitian laplacian. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 447–463. Springer, 2019.

[67] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[68] PE Gosden, AP MacGowan, and GC Bannister. Importance of air quality and related factors in the prevention of infection in orthopaedic implant surgery. *Journal of Hospital Infection*, 39(3):173–180, 1998.

[69] Sowjanya Gowrisankaran and James E Sheedy. Computer vision syndrome: A review. *Work*, 52(2): 303–314, 2015.

[70] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *International Conference on Machine Learning*, pages 1462–1471. PMLR, 2015.

[71] Jianhua Guo, Wei Huang, and Billy M Williams. Adaptive kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification. *Transportation Research Part C: Emerging Technologies*, 43:50–64, 2014.

[72] Shengnan Guo, Youfang Lin, Shijie Li, Zhaoming Chen, and Huaiyu Wan. Deep spatial–temporal 3d convolutional neural networks for traffic data forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3913–3926, 2019.

[73] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[74] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[75] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[76] Yixuan He, Gesine Reinert, Songchao Wang, and Mihai Cucuringu. Sssnet: Semi-supervised signed network clustering. In *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*, pages 244–252. SIAM, 2022.

[77] Yixuan He, Xitong Zhang, Junjie Huang, Mihai Cucuringu, and Gesine Reinert. Pytorch geometric signed directed: A survey and software on graph neural networks for signed and directed graphs. *arXiv preprint arXiv:2202.10793*, 2022.

[78] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[79] Joseph Hollingsworth, Brenna Copeland, and Jeremiah X Johnson. Are e-scooters polluters? The environmental impacts of shared dockless electric scooters. *Environmental Research Letters*, 14(8), 2019.

[80] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[81] Junjie Huang, Huawei Shen, Liang Hou, and Xueqi Cheng. Signed graph attention networks. In *International Conference on Artificial Neural Networks*, pages 566–577. Springer, 2019.

[82] Junjie Huang, Huawei Shen, Liang Hou, and Xueqi Cheng. Sdgnn: Learning node representation for signed directed networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(1):196–203, May 2021.

[83] IBM Research. Smart parking for improved Urban mobility. Accessed March. 27, 2023 [Online]. URL https://researcher.watson.ibm.com/researcher/view_group.php?id=9786.

[84] Invers. Global Moped Sharing Market Report 2022. Accessed March. 27, 2023 [Online]. URL https://go.invers.com/en/resources/global-moped-sharing-market-report-2022?utm_campaign=GMSMR+2022&utm_medium=mopedsharing.com&utm_source=referral.

[85] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[86] Sibren Isaacman, Richard Becker, Ramón Cáceres, Margaret Martonosi, James Rowland, Alexander Varshavsky, and Walter Willinger. Human mobility modeling at metropolitan scales. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 239–252, 2012.

[87] Toru Ishida. Digital city kyoto. *Communications of the ACM*, 45(7):76–81, 2002.

[88] Mohammad Raihanul Islam, B. Aditya Prakash, and Naren Ramakrishnan. Signet: Scalable embeddings for signed networks. In Dinh Phung, Vincent S. Tseng, Geoffrey I. Webb, Bao Ho, Mohadeseh Ganji, and Lida Rashidi, editors, *Advances in Knowledge Discovery and Data Mining*, pages 157–169, Cham, 2018. Springer International Publishing. ISBN 978-3-319-93037-4.

[89] Istituto Nazionale di Statistica. Basi Territoriali e variabili censuarie. Accessed March. 27, 2023 [Online]. URL https://www.istat.it/it/archivio/104317.

[90] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.

[91] Muhammad Aqib Javed, Muhammad Shahzad Younis, Siddique Latif, Junaid Qadir, and Adeel Baig. Community detection in networks: A multidisciplinary review. *Journal of Network and Computer Applications*, 108:87–111, 2018.

[92] Neal Jean, Sherrie Wang, Anshul Samar, George Azzari, David Lobell, and Stefano Ermon. Tile2vec: Unsupervised representation learning for spatially distributed data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3967–3974, 2019.

[93] Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey. *CoRR*, abs/2101.11174, 2021. URL https://arxiv.org/abs/2101.11174.

[94] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[95] K. Johnson. Environmental benefits of Smart City Solutions. Accessed March. 27, 2023 [Online]. URL https://www.climateforesight.eu/articles/environmental-benefits-of-smart-city-solutions/.

[96] Nal Kalchbrenner, Aäron Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. In *International Conference on Machine Learning*, pages 1771–1779. PMLR, 2017.

[97] Shiridhar Kashyap, Sudeep Kumar, Vikas Agarwal, Durga P Misra, Shubha R Phadke, and Aditya Kapoor. Protein protein interaction network analysis of differentially expressed genes to understand involved biological processes in coronary artery disease and its different severity. *Gene Reports*, 12: 50–60, 2018.

[98] Karen Kemp et al. *Encyclopedia of geographic information science*. Sage, 2008.

[99] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *Preprint submitted to arXiv arXiv:1609.02907*, 2016.

[100] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

[101] Nicos Komninos. The architecture of intelligent clities: Integrating human, collective and artificial intelligence to enhance knowledge and innovation. In *2006 2nd IET International Conference on Intelligent Environments-IE 06*, volume 1, pages 13–20. IET, 2006.

[102] Nikos Komodakis and Sergey Zagoruyko. Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017.

[103] Andrew Koster, Allysson Oliveira, Orlando Volpato, Viviane Delvequio, and Fernando Koch. Recognition and recommendation of parking places. In *Ibero-American Conference on Artificial Intelligence*, pages 675–685. Springer, 2014.

[104] Lauren J Krivo, Heather M Washington, Ruth D Peterson, Christopher R Browning, Catherine A Calder, and Mei-Po Kwan. Social isolation of disadvantage and advantage: The reproduction of inequality in urban space. *Social forces*, 92(1):141–164, 2013.

[105] Max Kuhn, Kjell Johnson, et al. *Applied predictive modeling*, volume 26. Springer, New York, 2013.

[106] Srijan Kumar, Francesca Spezzano, V. S. Subrahmanian, and Christos Faloutsos. Edge weight prediction in weighted signed networks. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 221–230, 2016.

[107] Stephen Law and Mateo Neira. An unsupervised approach to geographical knowledge discovery using street level and street network images. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery*, pages 56–65, 2019.

[108] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[109] Kyungeun Lee, Moonjung Eo, Euna Jung, Yoonjin Yoon, and Wonjong Rhee. Short-term traffic prediction with deep neural networks: A survey. *IEEE Access*, 9:54739–54756, 2021. doi: 10.1109/ACCESS.2021.3071174.

[110] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web*, pages 641–650, 2010.

[111] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Signed networks in social media. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1361–1370, 2010.

[112] John M Levy. *Contemporary urban planning*. Taylor & Francis, 2016.

[113] Wenxiang Li, Shawen Chen, Jieshuang Dong, and Jingxian Wu. Exploring the spatial variations of transfer distances between dockless bike-sharing systems and metros. *Journal of transport geography*, 92:103032, 2021.

[114] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

[115] Yanbing Li, Wei Zhao, and Huilong Fan. A spatio-temporal graph neural network approach for traffic flow prediction. *Mathematics*, 10(10):1754, 2022.

[116] Carlo Liberto, Gaetano Valenti, Silvia Orchi, Maria Lelli, Marialisa Nigro, and Marina Ferrara. The impact of electric mobility scenarios in large urban areas: The rome case study. *IEEE Transactions on Intelligent Transportation Systems*, 19(11):3540–3549, 2018.

[117] Elliott H Lieb and Michael Loss. Fluxes, Laplacians, and Kasteleyn's theorem. In *Statistical Mechanics*, pages 457–483. Springer, 1993.

[118] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

[119] Lingbo Liu, Ruimao Zhang, Jiefeng Peng, Guanbin Li, Bowen Du, and Liang Lin. Attentive crowd flow machines. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1553–1561, 2018.

[120] Q. Liu, Z. Liu, Y. Xu, W. Xiong, J. Yang, and Q. Wang. Toward identifying the urban community structure from population flow and public services distribution. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 3133–3142, 2020. doi: 10.1109/BigData50022.2020.9378225.

[121] Y. Liu, Z. Liu, C. Lyu, and J. Ye. Attention-based deep ensemble net for large-scale online taxi-hailing demand prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(11):4798–4807, 2020. doi: 10.1109/TITS.2019.2947145.

[122] Yang Liu, Cheng Lyu, Anish Khadka, Wenbo Zhang, and Zhiyuan Liu. Spatio-Temporal Ensemble Method for Car-Hailing Demand Prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(12):1–6, 2019. ISSN 1524-9050.

[123] Louisville Kentucky Open Data. Micro-mobility. Accessed March. 27, 2023 [Online]. URL https://data.louisvilleky.gov/pages/micro-mobility.

[124] Yang Lu and Li Da Xu. Internet of things (iot) cybersecurity research: A review of current research topics. *IEEE Internet of Things Journal*, 6(2):2103–2115, 2019. doi: 10.1109/JIOT.2018.2869847.

[125] Xiaolei Ma, Zhuang Dai, Zhengbing He, Jihui Ma, Yong Wang, and Yunpeng Wang. Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction. *Sensors*, 17(4):818, 2017.

[126] Catriona Manville, Gavin Cochrane, CAVE Jonathan, Jeremy Millard, Jimmy Kevin Pederson, Rasmus Kåre Thaarup, Andrea LIEBE WiK, and Matthias WISSNER WiK. Mapping smart cities in the eu. *EPRS: European Parliamentary Research Service*, 2014.

[127] J David Martin and Louis N Gray. Measurement of relative variation: Sociological examples. *American Sociological Review*, pages 496–502, 1971.

[128] Jeremy Miles. R-squared, adjusted r-squared. *Encyclopedia of statistics in behavioral science*, 2005.

[129] H Zare Moayedi and MA Masnadi-Shirazi. Arima model for network traffic prediction and anomaly detection. In *2008 International Symposium on Information Technology*, volume 4, pages 1–6. IEEE, 2008.

[130] Bojan Mohar. A new kind of hermitian matrices for digraphs. *Linear Algebra and its Applications*, 584: 343–352, 2020. ISSN 0024-3795.

[131] Seyed Sajad Mousavi, Michael Schukat, and Enda Howley. Deep reinforcement learning: An overview. In Yaxin Bi, Supriya Kapoor, and Rahul Bhatia, editors, *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016*, pages 426–440, Cham, 2018. Springer International Publishing. ISBN 978-3-319-56991-8.

[132] Marco De Nadai, Jacopo Staiano, Roberto Larcher, Nicu Sebe, Daniele Quercia, and Bruno Lepri. The death and life of great italian cities: A mobile phone data perspective. *Proceedings of the 25th International Conference on World Wide Web (WWW)*, pages 413–423, 2016.

[133] Taewoo Nam and Theresa A Pardo. Conceptualizing smart city with dimensions of technology, people, and institutions. In *Proceedings of the 12th annual international digital government research conference: digital government innovation in challenging times*, pages 282–291, 2011.

[134] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. *Balcan, Maria Florina and Weinberger, Kilian Q.*, 2016.

[135] M Nigro, S Peruzzi, C Liberto, and G Valenti. Urban-scale macroscopic fundamental diagram: an application to the real case study of rome. *Advances in Transportation Studies*, 42, 2017.

[136] NYC Taxi & Limousine Commission. TLC Trip Record Data. Accessed March. 27, 2023 [Online]. URL https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page.

[137] Aleksander Orlowski and Patrycja Romanowska. Smart cities concept: Smart mobility indicator. *Cybernetics and Systems*, 50(2):118–131, 2019.

[138] Shuai Pan, Lewis M Fulton, Anirban Roy, Jia Jung, Yunsoo Choi, and H Oliver Gao. Shared use of electric autonomous vehicles: Air quality and health impacts of future mobility in the united states. *Renewable and Sustainable Energy Reviews*, 149:111380, 2021.

[139] Enrica Papa and Dirk Lauwers. Smart mobility: Opportunity or threat to innovate places and cities. In *20th international conference on urban planning and regional development in the information society (REAL CORP 2015)*, pages 543–550, 2015.

[140] Wan-Joo Park, Byung-Sung Kim, Dong-Eun Seo, Dong-Suk Kim, and Kwae-Hi Lee. Parking space detection using ultrasonic sensor in parking assistance system. In *2008 IEEE intelligent vehicles symposium*, pages 1039–1044. IEEE, 2008.

[141] Janak Parmar, Pritikana Das, and Sanjaykumar M. Dave. Study on demand and characteristics of parking system in urban areas: A review. *Journal of Traffic and Transportation Engineering (English Edition)*, 7 (1):111–124, 2020. ISSN 2095-7564. Special Issue: Modeling and detecting traffic dynamics: granular, pedestrian and vehicular flow.

[142] Hao Peng, Bowen Du, Mingsheng Liu, Mingzhe Liu, Shumei Ji, Senzhang Wang, Xu Zhang, and Lifang He. Dynamic graph convolutional network for long-term traffic flow prediction with reinforcement learning. *Information Sciences*, 578:401–416, 2021. ISSN 0020-0255.

[143] Onel Pérez-Fernández and Juan Carlos García-Palomares. Parking places to moped-style scooter sharing services using gis location-allocation models and gps data. *ISPRS International Journal of Geo-Information*, 10(4):230, 2021.

[144] Laura W Perna. The benefits of higher education: Sex, racial/ethnic, and socioeconomic group differences. *The Review of Higher Education*, 29(1):23–52, 2005.

[145] Ate Poorthuis. How to draw a neighborhood? the potential of big data, regionalization, and community detection for understanding the heterogeneous nature of urban neighborhoods. *Geographical Analysis*, 50(2):182–203, 2018.

[146] Yan Qi and Sherif Ishak. A hidden markov model for short term prediction of traffic conditions on freeways. *Transportation Research Part C: Emerging Technologies*, 43:95–111, 2014.

[147] Xinwu Qian and Satish V. Ukkusuri. Spatial variation of the urban taxi ridership using gps data. *Applied Geography*, 59:31 – 42, 2015. ISSN 0143-6228.

[148] Navin Ranjan, Sovit Bhandari, Hong Zhao, Hoon Kim, and Pervez Khan. City-wide traffic congestion prediction based on cnn, lstm and transpose cnn. *IEEE Access*, PP:1–1, 04 2020. doi: 10.1109/ACCESS.2020.2991462.

[149] Mikko Rinne, Seppo Törmä, and D Kratinov. Mobile crowdsensing of parking space using geofencing and activity recognition. In *10th ITS European Congress, Helsinki, Finland*, pages 16–19, 2014.

[150] C Carl Robusto. The cosine-haversine formula. *The American Mathematical Monthly*, 64(1):38–40, 1957.

[151] Noe Samano, Mengjie Zhou, and Andrew Calway. You are here: Geolocation by embedding maps and images. In *European Conference on Computer Vision*, pages 502–518. Springer, 2020.

[152] Robert J Sampson. Neighbourhood effects and beyond: Explaining the paradoxes of inequality in the changing american metropolis. *Urban Studies*, 56(1):3–32, 2019.

[153] Aliaksei Sandryhaila and José MF Moura. Discrete signal processing on graphs. *IEEE transactions on signal processing*, 61(7):1644–1656, 2013.

[154] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 2483–2493. Curran Associates, Inc., 2018.

[155] Iqbal H Sarker. Data science and analytics: an overview from data-driven smart computing, decision-making and applications perspective. *SN Computer Science*, 2(5):1–22, 2021.

[156] Iqbal H. Sarker. Smart city data science: Towards data-driven smart cities with open research issues. *Internet of Things*, 19:100528, 2022. ISSN 2542-6605.

[157] Iqbal H Sarker and ASM Kayes. Abc-ruleminer: User behavioral rule-based machine learning method for context-aware intelligent services. *Journal of Network and Computer Applications*, 168:102762, 2020.

[158] Iqbal H Sarker, Alan Colman, Muhammad Ashad Kabir, and Jun Han. Individualized time-series segmentation for mining mobile phone user behavior. *The Computer Journal*, 61(3):349–368, 2018.

[159] Sanja Scepanovic, Sagar Joglekar, Stephen Law, and Daniele Quercia. Jane jacobs in the sky: Predicting urban vitality with open satellite data. *Proc. ACM Hum.-Comput. Interact.*, 5(CSCW1), apr 2021.

[160] Christian M Schneider, Vitaly Belik, Thomas Couronné, Zbigniew Smoreda, and Marta C González. Unravelling daily human mobility motifs. *Journal of The Royal Society Interface*, 10(84):20130246, 2013.

[161] John R Searle. Minds, brains, and programs. *Behavioral and brain sciences*, 3(3):417–424, 1980.

[162] Susan Shaheen, Nelson Chan, Apaar Bansal, and Adam Cohen. Shared mobility: A sustainability & technologies workshop: definitions, industry developments, and early understanding. 2015.

[163] Susan Shaheen, Adam Cohen, Nelson Chan, and Apaar Bansal. Sharing strategies: carsharing, shared micromobility (bikesharing and scooter sharing), transportation network companies, microtransit, and other innovative mobility modes. In *Transportation, land use, and environmental planning*, pages 237–262. Elsevier, 2020.

[164] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015.

[165] Donald C. Shoup. Cruising for parking. *Transport Policy*, 2006. ISSN 0967-070X.

[166] Yang Shuguan, Ma Wei, Pi Xidong, and Qian Sean. A deep learning approach to real-time parking occupancy prediction in spatio-temporal networks incorporating multiple spatio-temporal data sources. *Preprint submitted to Transportation Research Part C, Elsevier*, 2019.

[167] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[168] Stanislav Sobolevsky, Riccardo Campari, Alexander Belyi, and Carlo Ratti. General optimization technique for high-quality community detection in complex networks. *Physical Review E*, 90(1):012811, 2014.

[169] Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 914–921, 2020.

[170] Frances Sprei. Disrupting mobility. *Energy Research & Social Science*, 37:238–242, 2018. ISSN 2214-6296.

[171] Vincent Spruyt. Loc2vec: Learning location embeddings with triplet-loss networks. *Sentiance web article: https://www. sentiance. com/2018/05/03/venue-mapping*, 2018.

[172] Shiliang Sun, Changshui Zhang, and Guoqiang Yu. A bayesian network approach to traffic flow forecasting. *IEEE Transactions on intelligent transportation systems*, 7(1):124–132, 2006.

[173] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4278–4284. AAAI Press, 2017.

[174] Eiichi Taniguchi and Hiroshi Shimamoto. Intelligent transportation system based dynamic vehicle routing and scheduling with variable travel times. *Transportation Research Part C: Emerging Technologies*, 12 (3-4):235–250, 2004.

[175] Paula Tătulea, Florina Călin, Remus Brad, Lucian Brâncovean, and Mircea Greavu. An image feature-based method for parking lot occupancy. *Future Internet*, 11(8):169, 2019.

[176] Sean J Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.

[177] Miles Tight, Fiona Rajé, and Paul Timms. Car-free urban areas: A radical solution to the last mile problem or a step too far? *Built Environment*, 42(4):603–616, 2016.

[178] Leonardo Tolomei, Stefano Fiorini, Alessandro Ciociola, Luca Vassio, Danilo Giordano, and Marco Mellia. Benefits of relocation on e-scooter sharing - a data-informed approach. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 3170–3175, 2021.

[179] Yongxin Tong, Yuqiang Chen, Zimu Zhou, Lei Chen, Jie Wang, Qiang Yang, Jieping Ye, and Weifeng Lv. The simpler the better: A unified approach to predicting original taxi demands based on large-scale online platforms. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 1653–1662, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348874.

[180] Zekun Tong, Yuxuan Liang, Changsheng Sun, Xinke Li, David S. Rosenblum, and Andrew Lim. Digraph inception convolutional networks. *Advances in Neural Information Processing Systems*, 2020-December (NeurIPS):1–12, 2020. ISSN 10495258.

[181] Zekun Tong, Yuxuan Liang, Changsheng Sun, David S. Rosenblum, and Andrew Lim. Directed graph convolutional network, 2020.

[182] Zekun Tong, Yuxuan Liang, Henghui Ding, Yongxing Dai, Xinke Li, and Changhu Wang. Directed graph contrastive learning. *Advances in Neural Information Processing Systems*, 34:19580–19593, 2021.

[183] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[184] Tien Dung Tran, Nicolas Ovtracht, and Bruno Faivre d'Arcier. Modeling bike sharing system using built environment factors. *Procedia CIRP*, 30:293–298, 2015. ISSN 2212-8271. 7th Industrial Product-Service Systems Conference - PSS, industry transformation for sustainability and business.

[185] Eleni Tsironi, Pablo Barros, Cornelius Weber, and Stefan Wermter. An analysis of convolutional long short-term memory recurrent neural networks for gesture recognition. *Neurocomputing*, 268:76 – 86, 2017. Advances in artificial neural networks, machine learning and computational intelligence.

[186] TU Wienn. European Smart Cities 4.0 (2015). Accessed March. 27, 2023 [Online]. URL https://www.smart-cities.eu/?cid=01&ver=4.

[187] Sabine Kastner Ungerleider and Leslie G. Mechanisms of visual attention in the human cortex. *Annual review of neuroscience*, 23(1):315–341, 2000.

[188] United Nation Climate Change. The Paris Agreement. Accessed March. 27, 2023 [Online]. URL https://unfccc.int/process-and-meetings/the-paris-agreement/the-paris-agreement.

[189] United Nations Department of Economic and Social Affairs. 68% of the world population projected to live in urban areas by 2050, says UN. Accessed March. 27, 2023 [Online]. URL https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html.

[190] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[191] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[192] Eleni I Vlahogianni, Konstantinos Kepaptsoglou, Vassileios Tsetsos, and Matthew G Karlaftis. A real-time parking prediction system for smart cities. *Journal of Intelligent Transportation Systems*, 20 (2):192–204, 2016.

[193] Hongnian Wang and Han Su. Star: A concise deep learning framework for citywide human mobility prediction. In *2019 20th IEEE International Conference on Mobile Data Management (MDM)*, pages 304–309. IEEE, 2019.

[194] Jingyuan Wang, Jiawei Jiang, Wenjun Jiang, Chao Li, and Wayne Xin Zhao. Libcity: An open library for traffic prediction. In *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '21, page 145–148, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450386647.

[195] Mingshu Wang and Xiaolu Zhou. Bike-sharing systems and congestion: Evidence from us cities. *Journal of Transport Geography*, 65:147–154, 2017. ISSN 0966-6923.

[196] Pengyang Wang, Yanjie Fu, Jiawei Zhang, Xiaolin Li, and Dan Lin. Learning urban community structures: A collective embedding perspective with periodic spatial-temporal mobility graphs. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 9(6):1–28, 2018.

[197] Qi Wang, Nolan Edward Phillips, Mario L Small, and Robert J Sampson. Urban mobility and neighborhood isolation in america's 50 largest cities. *Proceedings of the National Academy of Sciences*, 115(30): 7735–7740, 2018.

[198] Suhang Wang, Jiliang Tang, Charu Aggarwal, Yi Chang, and Huan Liu. Signed network embedding in social media. In *Proceedings of the 2017 SIAM international conference on data mining*, pages 327–335. SIAM, 2017.

[199] Yingying Wang, Yibin Li, Yong Song, and Xuewen Rong. The influence of the activation function in a convolution neural network model of facial expression recognition. *Applied Sciences*, 10(5):1897, 2020.

[200] Yuan Wang, Chenwei Wang, Yinan Ling, Keita Yokoyama, Hsin-Tai Wu, and Yi Fang. Leveraging an efficient and semantic location embedding to seek new ports of bike share services. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1273–1282. IEEE, 2020.

[201] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[202] Zhecheng Wang, Haoyuan Li, and Ram Rajagopal. Urban2vec: Incorporating street view imagery and pois for multi-modal urban neighborhood embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1013–1020, 2020.

[203] Robert West, Hristo S Paskov, Jure Leskovec, and Christopher Potts. Exploiting social network structure for person-to-person sentiment analysis. *Transactions of the Association for Computational Linguistics*, 2:297–310, 2014.

[204] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[205] World Bank Open Data. Urban population (% of total population). Accessed March. 27, 2023 [Online]. URL https://data.worldbank.org/indicator/SP.URB.TOTL.IN.ZS.

[206] Chris Wortmann, Anne Magdalene Syré, Alexander Grahle, and Dietmar Göhlich. Analysis of electric moped scooter sharing in berlin: A technical, economic and environmental perspective. *World Electric Vehicle Journal*, 12(3), 2021. ISSN 2032-6653. doi: 10.3390/wevj12030096. URL https://www.mdpi.com/2032-6653/12/3/96.

[207] Chun-Hsin Wu, Jan-Ming Ho, and Der-Tsai Lee. Travel-time prediction with support vector regression. *IEEE transactions on intelligent transportation systems*, 5(4):276–281, 2004.

[208] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[209] Bo Xu, Ouri Wolfson, Jie Yang, Leon Stenneth, S Yu Philip, and Peter C Nelson. Real-time street parking availability estimation. In *2013 IEEE 14th International Conference on Mobile Data Management*, volume 1, pages 16–25. IEEE, 2013.

[210] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

[211] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[212] Ziru Xu, Yunbo Wang, Mingsheng Long, Jianmin Wang, and MOE KLiss. Predcnn: Predictive learning with cascade convolutions. In *IJCAI*, pages 2940–2947, 2018.

[213] Y. Zhang Y. Yin, Z. Liu and R. Zimmermann S. Wang, R. R. Shah. Gps2vec: Towards generating worldwide gps embeddings. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 416–419, 2019.

[214] Jihoon Yang, Jorge Portilla, and Teresa Riesgo. Smart parking service based on wireless sensor networks. In *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*, pages 6029–6034. IEEE, 2012.

[215] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, and Zhenhui Li. Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5668–5675, 2019.

[216] Xueyan Yin, Genze Wu, Jinze Wei, Yanming Shen, Heng Qi, and Baocai Yin. Deep learning on traffic prediction: Methods, analysis, and future directions. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4927–4943, 2022.

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

[218] Fengquan Yu, Jianhua Guo, Xiaobo Zhu, and Guogang Shi. Real time prediction of unoccupied parking space using time series model. In *2015 International conference on transportation information and safety (ICTIS)*, pages 370–374. IEEE, 2015.

[219] Rose Yu, Yaguang Li, Cyrus Shahabi, Ugur Demiryurek, and Yan Liu. Deep learning: A generic approach for extreme condition traffic forecasting. In *Proceedings of the 2017 SIAM international Conference on Data Mining*, pages 777–785. SIAM, 2017.

[220] Chen Yuan, Ye Li, Helai Huang, Shiqi Wang, Zhenhao Sun, and Yan Li. Using traffic flow characteristics to predict real-time conflict risk: A novel method for trajectory data analysis. *Analytic Methods in Accident Research*, 35:100217, 2022. ISSN 2213-6657.

[221] Jing Yuan, Yu Zheng, and Xing Xie. Discovering regions of different functions in a city using human mobility and pois. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 186–194, 2012.

[222] Jie Zhang, Bo Hui, Po-Wei Harn, Min-Te Sun, and Wei-Shinn Ku. smgc: A complex-valued graph convolutional network via magnetic laplacian for directed graphs, 2021.

[223] Junbo Zhang, Yu Zheng, Dekang Qi, Ruiyuan Li, Xiuwen Yi, and Tianrui Li. Predicting citywide crowd flows using deep spatio-temporal residual networks. *Artificial Intelligence*, 259:147–166, 2018.

[224] Xitong Zhang, Yixuan He, Nathan Brugnone, Michael Perlmutter, and Matthew Hirn. Magnet: A neural network for directed graphs, 2021.

[225] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3848–3858, 2019.

[226] Heliang Zheng, Jianlong Fu, Tao Mei, and Jiebo Luo. Learning multi-attention convolutional neural network for fine-grained image recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 5209–5217, 2017.

[227] Yanxu Zheng, Sutharshan Rajasegarar, and Christopher Leckie. Parking availability prediction for sensor-enabled car parks in smart cities. In *2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 1–6. IEEE, 2015.

[228] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3):1–55, 2014.

[229] Rui Zhu, Xiaohu Zhang, Dániel Kondor, Paolo Santi, and Carlo Ratti. Understanding spatio-temporal heterogeneity of bike-sharing and scooter-sharing mobility. *Computers, Environment and Urban Systems*, 81, 2020. ISSN 0198-9715.

[230] Yating Zhu, Xiaofei Ye, Jun Chen, Xingchen Yan, and Tao Wang. Impact of cruising for parking on travel time of traffic flow. *Sustainability*, 2020. ISSN 2071-1050.

[231] Xiaohan Zou. A survey on application of knowledge graph. In *Journal of Physics: Conference Series*, volume 1487, page 012016. IOP Publishing, 2020.

A

Appendix A

A.1 3D-CLoST: Impact of the Heuristic For Volume Construction

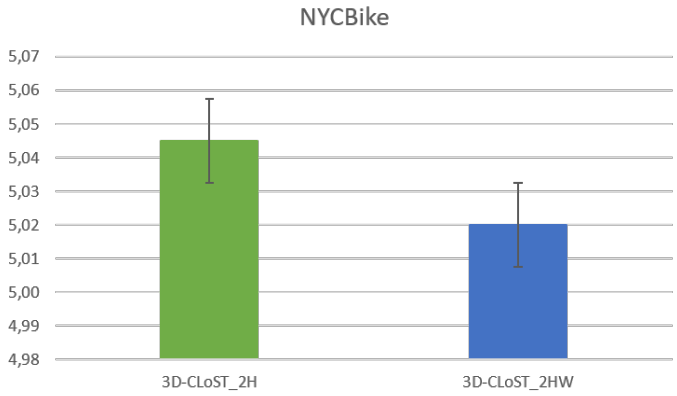


Figure A.1 Results of 3D-CLoST and its variants on NYCBike

To identify an optimized volume configuration for the two case studies, as explained in the Equation 4.3.1, we have started from the results extrapolated from the autocorrelation chart and explored the neighborhood of the resulted initial configuration to identify possible better volume composition. As regards the New York City dataset (with 1 hour time period), the volume suggested by the analysis of the autocorrelation chart is the $[F_{t-1}, F_{t-2}, F_{t-3}, F_{t-167}, F_{t-168}, F_{t-169}]$, while, for the data set of BJTaxi (with 30 minute time period) volume is composed of $[F_{t-1}, F_{t-2}, F_{t-46}, F_{t-47}, F_{t-48}, F_{t-49}, F_{t-50}]$.

Taking into consideration the periods (*closeness*, *recent* and *distant*) identified by the autocorrelation analysis, we have decreased the number of time intervals considered by 1, 2 and 3 hours for closeness period

and 1 - 6 for the other two periods. For example, the configuration with the maximum number of frames in the NYCBike dataset is given by $[F_{t-1},...,F_{t-6},F_{t-164},...,F_{t-172}]$.

The best configuration, based on the performance obtained, with regard to bicycles in New York is $[F_{t-1},F_{t-2},F_{t-168}]$ and $[F_{t-1},F_{t-2},F_{t-46},F_{t-47},F_{t-48},F_{t-49},F_{t-50}]$ for the taxis in Beijing. Interestingly, for the BJTaxi case study the best volume configuration coincides with the one suggested by the autocorrelation analysis, whereas for NYCBike the best results are achieved using a configuration contained in the initial one but with fewer frames.

Starting from the best configuration, we have separately assessed the impact of the various components on the prediction quality. As for New York, two different configurations are compared:

- **3D-CLoST_2H**: the volume entered as input is composed of $[F_{t-1},F_{t-2}]$
- **3D-CLoST_2HW**: the volume entered as input is composed of $[F_{t-1},F_{t-2},F_{t-168}]$

Similarly, for Beijing taxis:

- **3D-CLoST_D**: the volume entered as input is composed of $[F_{t-1},f_{t-2}]$
- **3D-CLoST_2HD**: the volume entered as input is composed of $[F_{t-1},F_{t-2},F_{t-46},F_{t-47},F_{t-48},F_{t-49},F_{t-50}]$

Notice that it is impossible to analyze the volume configuration consisting only of the distant (weekly) period in New York and only of closeness in Beijing without altering the kernels, whose depth coordinate are set to 2 and 3, respectively.

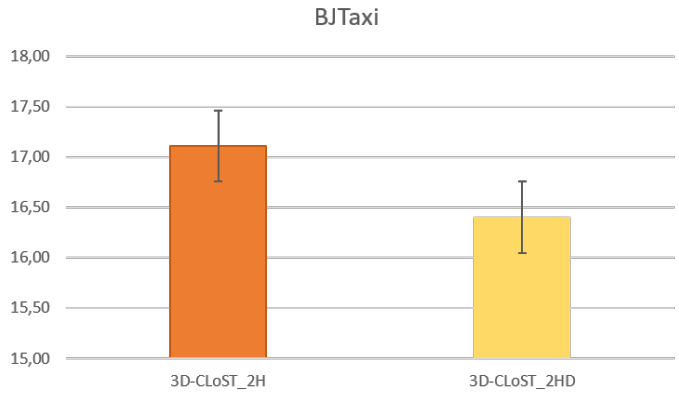


Figure A.2 Results of 3D-CLoST and its variants on BJTaxi

Figure A.1 shows how the 3D-CLoST_2HD configuration achieves the best performance (even if the difference is statistically significant only compared to 3D-CLoST_2H). This result seems to demonstrate that the use of all frames increases the performance capacity of the framework when the value of the information added compensates for the increased complexity of the model. Figure A.2 shows the comparison between the different volume configurations on the Beijing taxi dataset. As it can be seen from the graph, the 3D-CLoST_2HD configuration, i.e. the volume that brings together the frames of the previous hours and the previous day, reaches the lowest RMSE even if the difference with the RMSE of the configuration with only the hour frames previous to the hour to predict, is only marginally significant. This result highlights how the information contained by the closeness frames is the most important for a high-quality forecast, even if pieces of information from the daily period component can contribute to improving the overall performance.

A.2 STREED-Net: Ablation Study

In this section, an ablation study conducted on STREED-Net is presented, in which variations in the input structure and in the network architecture are analyzed. The study, for reasons of space, refers only to BikeNYC case study and does not involve the full combinatorics of all possible variants of the proposed model but aims to assess the impact on performance metrics of some parameters (namely, the number of input time points n) and specific architectural choices (viz., long skip connection, attention blocks, and external factors input branch), while maintaining all other conditions. More precisely, in what follows, STREED-Net is compared against the 5 different variations described below:

- **STREED-Net_N3**. Same architecture as STREED-Net, but input volumes with 3 frames $([X_{t-3}, X_{t-2}, X_{t-1})]$.
- **STREED-Net_N5**. Same architecture as STREED-Net, but input volumes with 5 frames $([X_{t-5}, X_{t-4}, X_{t-3}, X_{t-2}, X_{t-1}])$.
- **STREED-Net_NoLSC**. STREED-Net by removing the *long skip connection* between encoder and decoder.
- **STREED-Net_NoAtt**. STREED-Net without the *attention* blocks.
- **STREED-Net_NoExt**. STREED-Net without the external factors.

Notice that the study does not consider the variations with $n = 1$ and $n = 2$ as such values would not allow the network to capture meaningful temporal patterns between traffic flows.

Table A.1 reports the results of the ablation study conducted. Each data point in the table has been obtained performing 10 times the training procedure for each model variation changing the random seed, and evaluating the resulting network on the test set. The mean and standard deviation are reported.

Table A.1 Results obtained from ablation studies.

Model	RMSE	MAPE	APE
STREED-Net_N3	4.75 ± 0.04	21.18 ± 0.18	$3.28 \cdot 10^5 \pm 2.73 \cdot 10^3$
STREED-Net_N5	4.74 ± 0.03	21.03 ± 0.22	$3.26 \cdot 10^5 \pm 3.43 \cdot 10^3$
STREED-Net_NoLSC	4.84 ± 0.04	21.53 ± 0.24	$3.33 \cdot 10^5 \pm 3.71 \cdot 10^3$
STREED-Net_NoAtt	4.78 ± 0.04	20.95 ± 0.27	$3.25 \cdot 10^5 \pm 4.20 \cdot 10^3$
STREED-Net_NoExt	4.76 ± 0.04	20.99 ± 0.29	$3.26 \cdot 10^5 \pm 4.55 \cdot 10^3$
STREED-Net	**4.67 ± 0.03**	**20.85 ± 0.15**	$\mathbf{3.23 \cdot 10^5 \pm 2.31 \cdot 10^3}$

The results show that regarding the time horizon, for the BikeNYC case study, $n = 4$ allows the model to obtain better results. This means that, considering the particular setup, for the city of New York 4 hours of data allow to predict more accurately the dynamics of bicycle mobility whereas considering a greater amount of information ($n = 5$) would reduce the accuracy of the network. It is plausible to believe that considering a larger number of temporal instants would lead the network to grow in the number of parameters to be trained and thus require a larger amount of data to identify possible longer-term patterns.

From the architectural point of view, the two components attention block and long skip connection, confirm their importance in improving the performance of the proposed model, accounting for a 2.36% and 3.64%

increase in RMSE, respectively. In particular, as regards the attention block, not only STREED-Net reaches lower average error values, but also the standard deviation is reduced, proving that the attention blocks are effective in helping the network single out the most meaningful information and in making the training process more stable. Finally, the experiment shows also a strong impact of the long skip connection mechanism, which, as illustrated in Section 4.4.5, connects the encoder to the decoder to convey fine-grained details through the network.

B

Appendix B

B.1 Benefits of Relocation on E-scooter Sharing - a Data-Informed Approach

The cost of the system setup, the short lifetime of e-scooters, and the need for frequent battery charging operations call for system optimization to maximize fleet utilization, thus revenues [79]. To this extent, relocation policies play a fundamental role in optimizing the availability of e-scooters and satisfying users' mobility demands. Notice that relocation in the context of e-scooters has peculiar characteristics:

- A single worker can relocate multiple e-scooters at the same time.
- Given the typical short trip distance, customers look only for nearby e-scooters, making the spatial granularity much more fine-grained than for, e.g., car-sharing systems.
- The mobility demand is much more variable, given the more occasional usage of e-scooters [33, 229, 39].

This study examines, in terms of system performance and costs, the benefits of improving relocation by using a prediction system. What is the benefit in terms of mobility demand that the system can meet? Do these benefits bring additional profit? What is the importance of predicting e-scooters' shortages and surpluses accurately? To answer these questions, two models that predict the expected demand at a given time and place are trained. Then a real trace is produced using a simulator[1], comparing the performance of the system. In a nutshell, an observed rental demand at a given time is simulated. If an e-scooter exists nearby, it is rented and made available at the final location at the return time. If no scooter exists, on the other hand, it is recorded as an unfulfilled trip, i.e., a request from a user that cannot be fulfilled due to the lack of a vehicle. The system also simulates the battery charging process via battery swap. The case studies analyzed involve actual trips made available by the municipalities of Austin and Louisville.

[1] https://smartdata.polito.it/odysseus-an-origin-destination-simulator-of-shared-e-mobility-in-urban-scenarios/

Table B.2 Summary of system parameters and costs

Description	Var	Value
Scooter battery capacity	B	425 Wh [39]
Scooter efficiency	E_s	11 Wh/km [39]
Scooter cost (per year)	c_s	560 $/unit [a]
Unlock fee	f_0	1 $/trip [a]
Per minute fee	f_1	0.30 $/minute [a]
Relocation worker cost	c_w	15 $/hour [b]
Relocation vehicle cost	c_f	9.6 $/100 km [cd]
Relocation speed	w_s	20 km/h
Fleet size	N	variable
Number of relocation workers	n_w	variable

[a] *https://atommobility.com/blog-1/how-profitable-is-scooter-sharing-business*
[b] *https://www.indeed.com/cmp/Bird-Rides-Inc./salaries*
[c] *https://www.globalpetrolprices.com/USA/diesel_prices/*
[d] *https://www.fueleconomy.gov/feg/best/bestworstEPAtrucksNF.shtml*

B.2 Heuristic for scheduling relocations

Given the lists of *pick-up* and *drop-off* zones, it is necessary to define which relocation operations shall be implemented. This depends on the capacity of the system, e.g., the number of workers. Therefore, a simple greedy strategy is chosen to define which e-scooters to move from which zone to which zone.

First, each worker is associated to a single *pick-up* zone and to a single *drop-off* zone[2]. Iteratively, the *pick-up a* area with the largest positive *Delta* (i.e., the one with the largest expected abundance of e-scooters) and the *drop-off b* with the lowest negative *Delta* (i.e., the one with most predicted lack of e-scooters). Then the worker closest to the *pick-up* zone is identified and made to move a number of e-scooters equal to $min(\Delta(t,a), |\Delta(t,b)|, max\_capacity)$. The worker will then remain idle in the redeployment zone until the next redeployment program. $max\_capacity$ models the maximum number of e-scooters each worker can move, e.g., modeling the capacity of the support vehicle. To simplify the scenario, in the following, we set it very large and comment on this limit in the result section.

B.3 Datasets and Parameters

Table B.1 Dataset characteristics

City	N scooters	Avg trip dur.	Avg trip dist.	N zones	N trips train	N trips sim
Austin	8 350	899 s	1 288 m	2 794	4 642 309	527 776
Louisville	850	1 031 s	1 593 m	720	199 646	53 065

The train set, for both datasets, consists of data from August 2018 to August 2019. As a result, September 2019 trace was used for simulation and results collection. Dataset characteristics are summarized in Table B.1. As it

[2] This heuristic can be easily optimized by performing an actual path optimization based on relocation needs, with multiple pick-up and drop-off zones associated to a single worker.

can be seen, Austin e-scooter system is much bigger than Louisville. Moreover, this is reflected also in a much more heterogeneous temporal and spatial demand.

In the simulations, the following indexes are analyzed:

- **Satisfied demand.** Percentage of trips that are completed overall trip requests.
- **Marginal profit.** Revenues from satisfied trips minus costs of relocation and fleet. In this case, however, costs related to other aspects are not considered.

Table B.2 summarizes the system parameters that we keep fixed through all simulations, while we vary the fleet size N and number of workers n_w. Both affect the number of satisfied trips – thus the revenues – and costs.

Given the set of satisfied rentals $SatTrips$ in one month, revenues R_{tot} are:

$$R_{tot} = \sum_{i \in SatTrips} (f_0 + f_1 \cdot \delta t_i). \tag{B.1}$$

The costs C_{rel} for the relocation set $Rels$ account for the worker's and relocation vehicles costs:

$$C_{rel} = \sum_{j \in Rels} \left(\frac{c_w}{w_s} + c_f \right) [d(b_{j-1}, a_j) + d(a_j, b_j)] \tag{B.2}$$

where $d(a_j, b_j)$ is the distance between the pick-up and drop-off zone of relocation j, and $d(b_{j-1}, a_j)$ is the distance between worker's previous position and the next pick-up zone. Every worker is assigned a single relocation task every hour, so we take into consideration only the actual time to complete the relocation. As a result, the marginal profit for the month becomes:

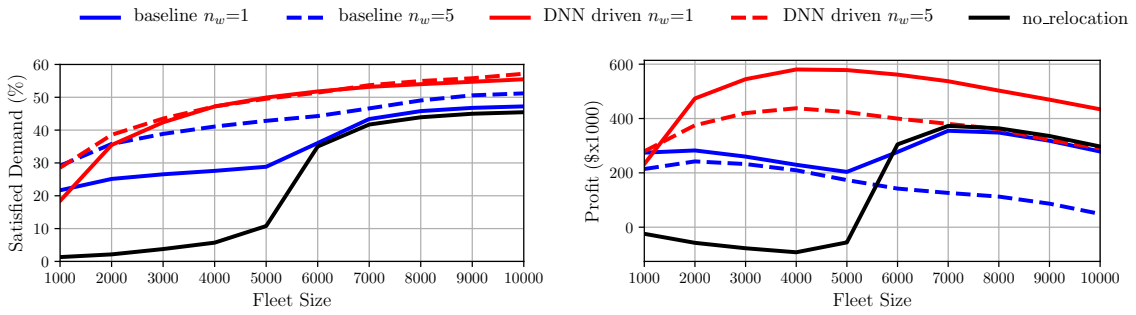
$$P = R_{tot} - C_{rel} - N \cdot c_s/12 \tag{B.3}$$

where $N \cdot c_s$ is the yearly cost of the e-scooter fleet. Notice that the chosen parameters (Table B.2) can have a significant influence on the revenues and marginal profit.

B.4 Extensive analysis on Austin and Louisville case studies

A detailed analysis of the two case studies is provided below.

B.4.1 Austin case study



(a) Satisfied demand

(b) Monthly marginal profit

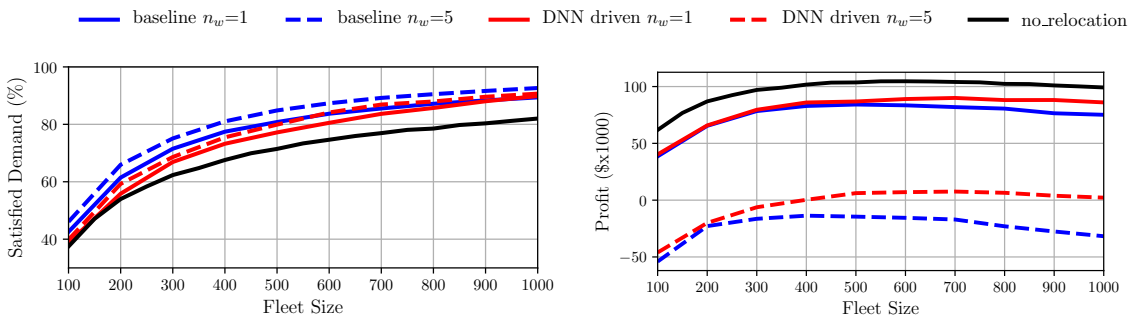
Figure B.1 Austin case study.

The metrics of interest (satisfied demand and monthly marginal profit) are evaluated by simulating different scenarios: the size of the N fleet and the predictive model used (baseline or DNN) are varied. Figure B.1a, shows the percentage of satisfied trips to Austin, setting $n_w = 1$ and $n_w = 5$ workers. The solid black curve shows the results without relocation. A system with a small fleet would not be able to meet user demand. With N greater than 6000, the system can meet about 40% of the demand, and increasing the number of e-scooters has little benefit.

Relocation significantly increases satisfied demand, especially for small fleets. Notice how with relocation, the same satisfied trip percentage can be obtained with a much smaller fleet size than without relocation. Intuitively, it is fundamental to move e-scooters where customers are looking for them. DNN offers the best results, improving by up to 42% the satisfied demand w.r.t. no relocation. Even for large fleet size, relocation allows 10% improvements in satisfied demand. Interestingly, $n_w = 1$ suffices with the accurate predictions offered by DNN, while the rough prediction based on averages requires more relocation operations every hour to see some benefits. One relocation per hour is already enough to improve system performance, provided the pick-up and drop-off zones are accurately predicted using the DNN model. This is confirmed by observing how many e-scooters are moved for each relocation/worker. With $N = 8000$ and $n_w = 1$, on average we move 28.4 vehicles with the DNN Driven predictions, while this reduces to 5.0 for the baseline ones. With $n_w = 5$ workers, the e-scooters moved for each relocation reduces to 10.2 for DNN and 3.1 for baseline. The additional workers move few e-scooters, bringing little overall benefits for DNN.

Figure B.1b shows the monthly marginal profit. For all systems, marginal profit increases with N when this is beneficial to improve the satisfied demand (left part of the figure). On the contrary, an excessive increase in fleet size increases costs, reducing profits (right side of figure). Focusing on $n_w = 1$ with DNN, the system results are always more profitable than a system with no relocation. That is, the extra-cost of relocation always pays-off in terms of additional revenues. With $N = 4000$, we move on average only 415 e-scooters per day. This allows a difference in marginal profit of 670 000 $ with respect to the case without relocation, for which we obtain a negative marginal profit. This is not true for the baseline model: as soon as $N > 5000$, the additional revenues are totally consumed by the relocation costs. With $n_w = 5$, revenues would reduce w.r.t. no relocation even for the DNN predictions, highlighting the need to accurately balance the benefits and costs of workers.

B.4.2 Louisville case study



(a) Satisfied demand

(b) Monthly marginal profit

Figure B.2 Louisville case study

Figure B.2a shows the percentage of satisfied requests in Louisville. Even without relocation, the satisfied demand grows up to 80% with 1000 scooters, and it increases by about an additional 10% with relocation. Here, the prediction based on the baseline works similarly to the DNN predictions, especially for $N > 600$. This happens because the system already performs quite well, and there are few areas that require a relocation operation. Regarding the marginal profit, Figure B.2b shows that the profit is maximum without relocation. This is because relocation results are too expensive compared to the extra revenues, and thus it is not sustainable with few trips in a city.

If the reader wishes to further explore the topic discussed in this appendix, they can refer to [178].

C

Appendix C

C.1 Properties of the Sign-Magnetic Laplacian

This section presents the proofs of theorems.

Theorem 1. *If $A \in \{0,1\}^{n\times n}$ and $q = 0.25$, we have $L^{\sigma} = L^{(q)}$.*

Proof. As $A \in \{0,1\}^{n\times n}$, D and $\bar{D}$ coincide. Let us consider H^{σ} and $H^{(q)}$. For each $i, j \in V$, if $A_{ij} = 1$, we have $H^{\sigma}_{ij} = -H^{\sigma}_{ji} = 0 + \mathbf{i}\frac{1}{2} = H^{(0.25)}_{ij} = -H^{(0.25)}_{ji}$; if $A_{ij} = 0$, we have $H^{\sigma}_{ij} = H^{\sigma}_{ji} = 1 + \mathbf{i}0 = H^{(0.25)}_{ij} = H^{(0.25)}_{ji}$. Thus, we have $L^{\sigma} = L^{(0.25)}$ and the claim follows. □

Theorem 2. *L^{σ} and L^{σ}_{norm} are positive semidefinite.*

Proof. Following the definition of the *Sign-Magnetic Laplacian*, we have $\Re(L^{\sigma}) = \bar{D}_s - A_s \odot (ee^{\top} - \mathrm{sgn}(|A - A^{\top}|))$ and $\Im(L^{\sigma}) = -A_s \odot \mathrm{sgn}(|A| - |A^{\top}|)$. As $\Re(L^{\sigma})$ is symmetric and $\Im(L^{\sigma})$ is skew symmetric by construction, L^{σ} is Hermitian. Since L^{σ} is Hermitian, $x^{*}\Im(L^{\sigma})x = 0$ holds for all $x \in \mathbb{C}^n$. As, by construction, $\bar{D}_s = \mathrm{Diag}(|A_s|\, e)$ and A_s is symmetric, the following holds for all $x \in \mathbb{C}^n$: $2x^{*}\Re(L^{\sigma})x$

$$
\begin{aligned}
&= 2\sum_{i,j=1}^{n} (\bar{D}_s)_{ij} x_i x_j^{*} - 2\sum_{i,j=1}^{n} (A_s)_{ij} x_i x_j^{*} (1 - \mathrm{sgn}(|A_{ij} - A_{ji}|)) \\
&= 2\sum_{i=1}^{n} (\bar{D}_s)_{ii} x_i x_i^{*} - 2\sum_{i,j=1}^{n} (A_s)_{ij} x_i x_j^{*} (1 - \mathrm{sgn}(|A_{ij} - A_{ji}|)) \\
&= 2\sum_{i,j=1}^{n} |(A_s)_{ij}||x_i|^2 - 2\sum_{i,j=1}^{n} (A_s)_{ij} x_i x_j^{*} (1 - \mathrm{sgn}(|A_{ij} - A_{ji}|)) \\
&= \sum_{i,j=1}^{n} |(A_s)_{ij}||x_i|^2 + \sum_{i,j=1}^{n} |(A_s)_{ji}||x_j|^2 \\
&- 2\sum_{i,j=1}^{n} (A_s)_{ij} x_i x_j^{*} (1 - \mathrm{sgn}(|A_{ij} - A_{ji}|))
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{i,j=1}^{n} |(A_s)_{ij}||x_i|^2 + \sum_{i,j=1}^{n} |(A_s)_{ij}||x_j|^2 \\
&- 2\sum_{i,j=1}^{n} (A_s)_{ij} x_i x_j^* \left(1 - \operatorname{sgn}(|A_{ij} - A_{ji}|)\right) \\
&= \sum_{i,j=1}^{n} |(A_s)_{ij}||x_i|^2 + \sum_{i,j=1}^{n} |(A_s)_{ij}||x_j|^2 \\
&- 2\sum_{i,j=1}^{n} |(A_s)_{ij}| \operatorname{sgn}((A_s)_{ij}) x_i x_j^* \left(1 - \operatorname{sgn}(|A_{ij} - A_{ji}|)\right) \\
&= \sum_{i,j=1}^{n} |(A_s)_{ij}| \Big(|x_i|^2 + |x_j|^2 - 2\operatorname{sgn}(A_{ij}) x_i x_j^* \left(1 - \operatorname{sgn}(|A_{ij} - A_{ji}|)\right) \Big) \\
&\geq \sum_{i,j=1}^{n} |(A_s)_{ij}| \Big(|x_i| - \operatorname{sgn}(A_{ij}) |x_j| \Big)^2 \\
&\geq 0.
\end{aligned}
$$

Thus, L^σ is positive semidefinite. Let us now consider the *normalized Sign-Magnetic Laplacian*, which, according to Eq. (5.4), is defined as $L^\sigma_{\text{norm}} = \bar{D}_s^{-\frac{1}{2}} L^\sigma \bar{D}_s^{-\frac{1}{2}}$. We need to show that $x^* L^\sigma_{\text{norm}} x \geq 0$ for all $x \in \mathbb{C}^n$. Letting $y = \bar{D}_s^{-\frac{1}{2}} x$, we have $x^* L^\sigma_{\text{norm}} x = x^* \bar{D}_s^{-\frac{1}{2}} L^\sigma \bar{D}_s^{-\frac{1}{2}} x = y^* L^\sigma y$, which is nonnegative as proven before. □

Theorem 3. $\lambda_{max}(L^\sigma_{norm}) \leq 2$.

Proof. Let $B := \bar{D}_s + H^\sigma$. Let us show that B is positive semidefinite. As B is Hermitian by construction, we have $x^* \Im(L^\sigma) x = 0$. Next, we show that $2x^* \Re(B) x \geq 0$.

$2x^* \Re(B) x$

$$
\begin{aligned}
&= 2\sum_{i,j=1}^{n} (\bar{D}_s)_{ij} x_i x_j^* + 2\sum_{i,j=1}^{n} (A_s)_{ij} x_i x_j^* \left(1 - \operatorname{sgn}(|A_{ij} - A_{ji}|)\right) \\
&= 2\sum_{i=1}^{n} (\bar{D}_s)_{ii} x_i x_i^* + 2\sum_{i,j=1}^{n} (A_s)_{ij} x_i x_j^* \left(1 - \operatorname{sgn}(|A_{ij} - A_{ji}|)\right) \\
&= 2\sum_{i,j=1}^{n} |(A_s)_{ij}||x_i|^2 + 2\sum_{i,j=1}^{n} (A_s)_{ij} x_i x_j^* \left(1 - \operatorname{sgn}(|A_{ij} - A_{ji}|)\right) \\
&= \sum_{i,j=1}^{n} |(A_s)_{ij}||x_i|^2 + \sum_{i,j=1}^{n} |(A_s)_{ji}||x_j|^2 \\
&+ 2\sum_{i,j=1}^{n} (A_s)_{ij} x_i x_j^* \left(1 - \operatorname{sgn}(|A_{ij} - A_{ji}|)\right) \\
&= \sum_{i,j=1}^{n} |(A_s)_{ij}||x_i|^2 + \sum_{i,j=1}^{n} |(A_s)_{ij}||x_j|^2 \\
&+ 2\sum_{i,j=1}^{n} (A_s)_{ij} x_i x_j^* \left(1 - \operatorname{sgn}(|A_{ij} - A_{ji}|)\right) \\
&= \sum_{i,j=1}^{n} |(A_s)_{ij}||x_i|^2 + \sum_{i,j=1}^{n} |(A_s)_{ij}||x_j|^2 \\
&+ 2\sum_{i,j=1}^{n} |(A_s)_{ij}| \operatorname{sgn}((A_s)_{ij}) x_i x_j^* \left(1 - \operatorname{sgn}(|A_{ij} - A_{ji}|)\right)
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{i,j=1}^{n} |(A_s)_{ij}| \Big(|x_i|^2 + |x_j|^2 + 2\,\mathrm{sgn}(A_{ij}) x_i x_j^* \left(1 - \mathrm{sgn}(|A_{ij} - A_{ji}|)\right) \Big) \\
&\geq \sum_{i,j=1}^{n} |(A_s)_{ij}| \Big(|x_i|^2 + |x_j|^2 \Big) \\
&\geq 0.
\end{aligned}
$$

Thus, the normalized version of B satisfies

$$x^* B_{\text{norm}} x = x^* \bar{D}_s^{-\frac{1}{2}} B \bar{D}_s^{-\frac{1}{2}} x = y^* B y \geq 0.$$

We have proved that $x^* B_{\text{norm}} x$ is positive semidefinite. Hence, the following holds:

$$
\begin{aligned}
& x^* B_{\text{norm}} x \geq 0 \\
& x^* \left(I + D^{-\frac{1}{2}} H^{\sigma} D^{-\frac{1}{2}} \right) x \geq 0 \\
& - x^* D^{-\frac{1}{2}} H^{\sigma} D^{-\frac{1}{2}} x \leq x^* x \\
& x^* I x - x^* D^{-\frac{1}{2}} H^{\sigma} D^{-\frac{1}{2}} x \leq 2 x^* x \\
& \frac{x^* L^{\sigma}_{\text{norm}} x}{x^* x} \leq 2.
\end{aligned}
$$

Due to the Courant-Fischer theorem applied to L^{σ}_{norm}, we have:

$$\lambda_{\max} = \max_{x \neq 0} \frac{x^* L^{\sigma}_{\text{norm}} x}{x^* x}.$$

Thus, $\lambda_{\max} \leq 2$ holds. □

Theorem 4. *Given a constant $\alpha \in \mathbb{R}^+$, L^{σ} satisfies the following positive homogeneity property:*

$$L^{\sigma}(\alpha A) = \alpha L^{\sigma}(A),$$

where $L^{\sigma}(\alpha A)$ and $L^{\sigma}(A)$ are the Sign-Magnetic Laplacian matrices of a directed graph with, respectively, adjacency matrix $\alpha A \in \mathbb{R}^{n \times n}$ and $A \in \mathbb{R}^{n \times n}$.

Proof. Let $H^{\sigma}(X)$ and $\bar{D}(X)$ be the H^{σ} and $\bar{D}$ matrices of a directed graph with adjacency matrix $X \in \mathbb{R}^{n \times n}$. We have:

$H^{\sigma}(A \odot B)$

$$
\begin{aligned}
&= \left(\frac{A \odot B + A^{\top} \odot B^{\top}}{2} + \mathbf{i} \frac{A \odot B + A^{\top} \odot B^{\top}}{2} \right) \odot \\
&\left((ee^{\top} - \mathrm{sgn}(|A - A^{\top}|)) + \mathbf{i}\, \mathrm{sgn}(|A| - |A^{\top}|) \right) = \\
&= \left(\alpha \frac{(A + A^{\top})}{2} + \mathbf{i} \frac{\alpha (A + A^{\top})}{2} \right) \odot \\
&\left((ee^{\top} - \mathrm{sgn}(|A - A^{\top}|)) + \mathbf{i}\, \mathrm{sgn}(|A| - |A^{\top}|) \right) = \\
&= \left(\frac{A + A^{\top}}{2} + \mathbf{i} \frac{A + A^{\top}}{2} \right) \odot
\end{aligned}
$$

$$\Big((ee^\top - \operatorname{sgn}(|A-A^\top|)) + \mathbf{i}\operatorname{sgn}(|A|-|A^\top|)\Big)\odot B = \\ = H^\sigma(A)\odot B.$$

$\bar{D}(A\odot B) = \operatorname{Diag}(|A\odot B|e) = \operatorname{Diag}(|A|e) = \bar{D}(A)\odot B$ (the latter by construction of B). The claim follows. □

Theorem 5. *Consider a weighted directed graph $G=(V,E)$ without pairs of antiparallel edges (digons). Given a directed edge $(i,j)\in E$ of weight w_{ij}, let $G'=(V,E)$ be a copy of G obtained by reversing the direction of (i,j) into (j,i) and flipping the sign of its weight by letting $w_{ji}=-w_{ij}$. Let $L^\sigma(G)$ and $L^\sigma(G')$ be the L^σ matrix defined on G and G', respectively. $L^\sigma(G)=L^\sigma(G')$ holds.*

Proof. Let A_G and $A_{G'}$ be the adjacency matrices of G and G'. Let $H^\sigma(X)$ and $A_s(X)$ be the H^σ and A_s matrices defined for a graph with adjacency matrix X. $\Re(H^\sigma(G))=\Re(H^\sigma(G'))$ holds since both $A_G-A_G^\top$ and $A_{G'}-A_{G'}^\top$ are nonzero in positions i,j and j,i and, thus, $(ee^\top-\operatorname{sgn}(|A_G-A_G^\top|))$ and $(ee^\top-|\operatorname{sgn}(A_{G'}-A_{G'}^\top)|)$ are both equal to 0 in these positions. To see that $\Im(H^\sigma(G))=\Im(H^\sigma(G'))$, we observe that $A_s(A_G)_{ij}=-A_s(A_{G'})_{ij}$ and $A_s(A_G)_{ji}=-A_s(A_{G'})_{ji}$, but also that $\operatorname{sgn}(|A_G|-|A_G^\top|)_{ij}=-\operatorname{sgn}(|A_{G'}|-|A_{G'}^\top|)_{ij}$ and that $\operatorname{sgn}(|A_G|-|A_G^\top|)_{ji}=-\operatorname{sgn}(|A_{G'}|-|A_{G'}^\top|)_{ji}$. Thus, the two differences in sign cancel out and the claim follows. □

Further observation The results presented in this work still hold if the imaginary part of H^σ is multiplied by any nonnegative real constant $\varepsilon>0$. If $A\in\{0,1\}^{n\times n}$, by choosing $\varepsilon=\sqrt{3}$, L^σ coincides with the Hermitian matrix "of the second kind" proposed in [130] in the context of algebraic graph theory.

C.2 Sign-pattern inconsistency of $L^{(q)}$

We highlighted that the *Magnetic Laplacian*, $L^{(q)}$, exhibits a crucial sign-pattern inconsistency. Indeed, while, for unweighted graphs, $L^{(q)}$ encodes the directional information of the edges in the sign of the imaginary part of $H^{(q)}$, this is not necessarily the case for weighted graphs as the sign pattern of $H^{(q)}$ can change drastically by just scaling the weights of the graph by a positive constant.

To better illustrate this, we introduce the following example. Consider a directed graph $G=(V,E)$ with $V=\{1,2\}$ and $E=\{(1,2)\}$. Let us assume that the weight of the $(1,2)$ edge can take one of the following four values: 0.8, 2, 5, 36 and let $q=0.25$. Although the direction of the edge $(1,2)$ does not change, based on the magnitude of the weight, we observe four different scenarios.

1. $A=\begin{bmatrix}0 & 0.8\\ 0 & 0\end{bmatrix}, A_s=\begin{bmatrix}0 & 0.4\\ 0.4 & 0\end{bmatrix}$, and $H^{(0.25)}=\begin{bmatrix}0 & 0.4\\ 0.4 & 0\end{bmatrix}\odot\begin{bmatrix}1 & 0.31\\ 0.31 & 1\end{bmatrix}+\mathbf{i}\begin{bmatrix}0 & 0.40\\ 0.40 & 0\end{bmatrix}\odot\begin{bmatrix}0 & 0.95\\ -0.95 & 0\end{bmatrix}$.
   We have $\operatorname{sgn}(\Im(H^{(0.25)})_{12})=-\operatorname{sgn}(\Im(H^{(0.25)})_{21})=\operatorname{sgn}(A_{12})$ and, thus, the sign of the imaginary part of $H^{(0.25)}$ encodes the direction of the edge, while $\Re(H^{(0.25)})_{12}=\Re(H^{(0.25)})_{21}\neq 0$.
2. $A=\begin{bmatrix}0 & 2\\ 0 & 0\end{bmatrix}$, $A_s=\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$, and $H^{(0.25)}=\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\odot\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}+\mathbf{i}\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\odot\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}$. We have $\Im(H^{(0.25)})_{12}=\Im(H^{(0.25)})_{21}=0$ and, thus, the sign of the imaginary part of $H^{(0.25)}$ does not encode at all the direction of the edge. Furthermore, we note that $\operatorname{sgn}(\Re(H^{(0.25)})_{12})=\operatorname{sgn}(\Re(H^{(0.25)})_{21})\neq\operatorname{sgn}(A_{12})$. Consequently, the matrix $H^{(0.25)}$ represents the graph as an undirected graph with a negative weight.

3. $A = \begin{bmatrix} 0 & 5 \\ 0 & 0 \end{bmatrix}$, $A_s = \begin{bmatrix} 0 & 2.5 \\ 2.5 & 0 \end{bmatrix}$, and $H^{(0.25)} = \begin{bmatrix} 1 & 2.5 \\ 2.5 & 1 \end{bmatrix} \odot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \mathbf{i} \begin{bmatrix} 0 & 2.5 \\ 2.5 & 0 \end{bmatrix} \odot \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$. We have $\operatorname{sgn}(\Im(H^{(0.25)})_{12}) = -\operatorname{sgn}(\Im(H^{(0.25)})_{21}) = \operatorname{sgn}(A_{12})$; thus, the sign of the imaginary part of $H^{(0.25)}$ encodes the direction of the edge $(1,2)$ consistently with A, while $\Re(H^{(0.25)})_{12} = \Re(H^{(0.25)})_{21} = 0$;
4. $A = \begin{bmatrix} 0 & 36 \\ 0 & 0 \end{bmatrix}$, $A_s = \begin{bmatrix} 0 & 18 \\ 18 & 0 \end{bmatrix}$, and $H^{(0.25)} = \begin{bmatrix} 0 & 18 \\ 18 & 0 \end{bmatrix} \odot \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} + \mathbf{i} \begin{bmatrix} 0 & 18 \\ 18 & 0 \end{bmatrix} \odot \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$. We have $\Im(H^{(0.25)})_{12} = \Im(H^{(0.25)})_{21} = 0$ and, thus, the sign of the imaginary part of $H^{(0.25)}$ does not encode the direction of the edge, while $\operatorname{sgn}(\Re(H^{(0.25)})_{12}) = \operatorname{sgn}(\Re(H^{(0.25)})_{21}) = \operatorname{sgn}(A_{12})$. Consequently, the matrix $H^{(0.25)}$ represents the graph as an undirected graph with a positive weight.

C.3 Flow-based pre-processing

If applied an edge at a time, Theorem 5 can be used to transform a given directed graph with digons into a multigraph. For applications where the graph entails a flow-like relationship, it is then natural to aggregate every pair of parallel edges thus obtained into a single edge by summing their weights, thereby obtaining a (simple) weighted graph. In more details, consider two antiparallel edges (i,j) and (j,i) with different weights $(w_{ij} \neq w_{ji})$. By applying Theorem 5 to the (i,j) arc, we reverse its direction into (j,i) and flip the sign of its weight, thus obtaining the edge (j,i) of weight $w_{ji} := -w_{ij}$. As the graph already contains an (j,i) arc, the graph is turned into a multigraph. If the graph models a flow-like relationship, it is reasonable to collapse such a pair of parallel edges into a single edge of weight equal to $w_{ji} := -w_{ij} + w_{ji}$. We carry out this operation as a pre-processing activity for each task except for the link sign prediction task, whose datasets do not entail flow-like information.

In the following, we report a quantitative example to show the positive impact of this technique. In more detail, we consider two scenarios:

1. The flow-based pre-processing is not applied to the graph. As a consequence, some information related to the topology of the graph is lost.
2. The flow-based pre-processing is applied to the graph. No information is lost.

Consider a graph with a pair of antiparallel edges represented by the adjacency matrix $A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$.

1. If we do not apply the flow-based pre-processing, we have $A_s = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$. Thus, L^{σ} (but also $L^{(q)}$) fails to represent the graph as the entire pair of antiparallel edges is lost.
2. If we apply the flow-based pre-processing to the graph (not applicable for $L^{(q)}$), we obtain the following new adjacency matrix: $A_{\text{new}} = \begin{bmatrix} 0 & 2 \\ 0 & 0 \end{bmatrix}$; thus, we have $A_{s_{\text{new}}} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. Thanks to this, L^{σ} consistently represent a graph with one edge, the direction of which is encoded in the imaginary part of L^{σ}.

C.4 Details on Datasets

All the datasets we considered can be obtained from our code.

Real-world datasets. We test SigMaNet on six real-world datasets: `Bitcoin-OTC` and `Bitcoin Alpha` [106]; `Slashdot` and `Epinions` [111]; `WikiRfa` [203]; and `Telegram` [26]. The first two datasets, `Bitcoin-OTC` and `Bitcoin Alpha`, come from exchange operations: Bitcoin-OTC and Bitcoin Alpha. Both of these exchanges allow users to rate the others on a scale of -10 to $+10$ (excluding 0). According to the OTC guidelines, scammers should be given a score of -10, while at the other end of the spectrum, $+10$ means full trust. Other evaluation values have intermediate meanings. Therefore, these two exchanges explicitly lead to a graph with weights unrestricted in sign. The other two datasets are `Slashdot` and `Epinions`. The first comes from a tech news website with a community of users. The website introduced Slashdot Zoo features that allow users to tag each other as friend or foe. The dataset represents a signed social network with friend $(+1)$ and enemy (-1) labels. `Epinions` is an online who-trust-who social network of a consumer review site (*Epinions.com*). Site members can indicate their trust or distrust of other people's reviews. The network reflects people's views on others. `WikiRfa` is a collection of votes given by Wikipedia members collected from 2003 to 2013. Indeed, any Wikipedia member can vote for support, neutrality, or opposition to a Wikipedia editor's nomination for administrator. This leads to a directed, multigraph (unrestricted in sign) in which nodes represent Wikipedia members and edges represent votes, which is then transformed into a simple graph by condensing any parallel edges into a single edge of weight equal to the sum of the weights of the original edges. The graph features a higher number of nodes and edges than the one proposed in [82]. In these five datasets, the classes of positive and negative edges are imbalanced (see Table C.1). The last dataset is `Telegram`, an influence network that analyses the interactions and influences between distinct groups and actors who associate and propagate political ideologies. This is a pairwise-influence network between 245 Telegram channels with 8912 links. The labels are generated following the method discussed in [26], with a total of four classes.

Table C.1 Statistics of the six datasets

Data set	n	$\lvert\varepsilon^+\rvert$	$\lvert\varepsilon^-\rvert$	% pos	Directed	Weighted	Density
`Telegram`	245	8,912	0	100.00	✓	✓	14.91%
`Bitcoin-Alpha`	3,783	22,650	1,536	93.65	✓	✓	0.17%
`Bitcoin-OTC`	5,881	32,029	3,563	89.99	✓	✓	0.10%
`WikiRfA`	11,381	138,143	39,038	77.97	✓	✓	0.14%
`Slashdot`	82,140	425,072	124,130	77.70	✓	×	0.01%
`Epinion`	131,828	717,667	123,705	85.30	✓	×	0.01%

Synthetic dataset. The synthetic set of graphs are generated via a direct stochastic block model (DSBM) with (unlike in [224]) edge weights in the range $\mathbb{N} \cap [2, 1000]$. In detail, in DSBM we define a number of nodes n and a number of clusters C which partition the vertices into communities of equal size. We define a collection of probabilities $\{\alpha_{ij}\}_{1 \le i,j \le C}$, where $0 \le \alpha_{ij} \le 1$ with $\alpha_{ij} = \alpha_{ji}$, to define the probability that an undirected edge be generated between a node u and a node v that belong to two different clusters, i.e., $u \in C_i$ and $v \in C_j$, and α_{ii} is the probability that an undirected edge is generated between two nodes in the same cluster. As the generated graph is undirected, we follow [224] and introduce a rule to transform the graph from undirected to directed: we define a collection of probabilities $\{\beta_{ij}\}_{1 \le i,j \le C}$, where $0 \le \beta_{ij} \le 1$ such that $\beta_{i,j} + \beta_{j,i} = 1$. Each edge $\{u, v\}$ is assigned a direction using the rule that the edge points from u to v with probability β_{ij} if $u \in C_i$ and $v \in C_j$, and points from v to u with probability β_{ji}. For the characteristics of the loss function present in the SSSNET model, we set 10% of the nodes per class of the graph as seed nodes.

C.5 Experiment Details

Hardware. The experiments were conducted on 2 different computers: one with 1 NVIDIA Tesla T4 GPU, 380 GB RAM, and Intel(R) Xeon(R) Gold 6238R CPU @ 2.20GHz CPU, and the other with 1 NVIDIA TITAN Xp GPU, 80 GB RAM, and Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz CPU.

Model Settings. We train all the models considered in this work with a maximum of 3000 epochs and early stop if the validation error does not decrease after 500 epochs for both node classification and link prediction tasks. As in [224], one dropout layer with a probability of 0.5 is created before the last layer. We set the parameter $K = 1$ for ChebNet, MagNet, and SigMaNet. A hyperparameter optimization procedure is adopted to identify the best set of parameters for each model. We tune the number of filters in $\{16, 32, 64\}$ for the graph convolutional layers for all models except for DGCN. We set for both node classification and link prediction a learning rate of 10^{-3}. For link sign prediction task, the learning rate is set in $\{10^{-2}, 5 \cdot 10^{-3}, 10^{-3}\}$. We employ Adam as the optimization algorithm, and set weight decays (regularization hyperparameter) to $5 \cdot 10^{-4}$ to prevent overfitting.

Some further details are reported in the following:

- The coefficient q for MagNet is chosen in $\{0.01, 0.05, 0.1, 0.15, 0.2, 0.25\}$.
- The coefficient α for PageRank-based models (APPNP and DiGraph) is chosen in $\{0.05, 0.1, 0.15, 0.2\}$.
- For APPNP, we set $K = 10$ for node classification (parameter suggested in [100]), and select K in $\{1, 5, 10\}$ for link prediction.
- For GAT, we adopt a number of heads in $\{2, 4, 8\}$.
- DGCN is somewhat different from the other networks because it requires generating three matrices of order proximity, i.e., first-order proximity, second-order *in-degree proximity* and second-order *out-degree proximity*. For this network, the number of filters for each channel is searched in $\{5, 15, 30\}$ for node classification and link prediction.
- In GIN, the parameter ε is set to 0 for both tasks.
- In SSSNET, parameters γ_s and γ_t are set to 50 and 0.1 respectively.
- In ChebNet and GCN, the symmetrized adjacency matrix $A_s = \frac{A+A^\top}{2}$ is used.
- For DiGCL, we select the *Pacing function* in [*linear*, *exponential*, *logarithmic*, *fixed*]. We also adopt two different configuration: *i)* $\tau = 0.4$, *drop feature rate 1* = 0.3 and *drop feature rate 2* = 0.4, and *ii)* $\tau = 0.9$, *drop feature rate 1* = 0.2 and *drop feature rate 2* = 0.1.

Link prediction. In these tasks, we define the feature matrix $X \in \mathbb{R}^{n \times 2}$ in such a way that, for each node $i \in V$, X_{i1} is the in-degree of node i and X_{i2} is the node's out-degree. This is done to allow the models to learn structural information directly from the adjacency matrix. In particular, for the sign link prediction task, we use in-degree and out-degree by computing the absolute value of their edge weights.

Node classification. In this task, for the `Telegram` dataset we retain the dataset's original features, whereas, for the synthetic datasets, we create them via the in-degree and out-degree vector as explained before.

C.6 Complexity of SigMaNet

Assuming, as done in our experiments, that SigMaNet features two graph-convolutional layers with f_1 and f_2 filters, each defined as in Equation (5.5) and c features per node, the complexity of SigMaNet is $O(nc(n+f_1)+nf_1(n+f_2)+m^{\text{train}}f_2d)$ for link prediction (sign/direction/existence task) and $O(nc(n+f_1)+nf_1(n+f_2)+nf_2d))$ for node classification, where m^{train} is the number of edges in the training set and d is the number of classes. The detailed calculations for complexity are as follows:

1. The equation $\tilde{D}^{-1/2}H^{\sigma}\tilde{D}^{-1/2} \in \mathbb{C}^{n\times n}$ is computed in $O(|H^{\sigma}|) = O(n^2)$ in pre-processing (once per graph, independently of the node features).

2. The first convolutional layer requires, $O(n^2c+ncf_1+nf_1) = O(nc(n+f_1))$ due to 3 operations:

   (a) It multiplies $\tilde{D}^{-1/2}H^{\sigma}\tilde{D}^{-1/2}$ by the node-feature matrix $X \in \mathbb{C}^{n\times c}$, obtaining $P^{11} \in \mathbb{C}^{n\times c}$ in $O(n^2c)$ (assuming matrix multiplications require cubic time);

   (b) It multiplies P^{11} by the weight matrix $\Theta \in \mathbb{R}^{c\times f_1}$, obtaining $P^{12} \in \mathbb{C}^{n\times f_1}$ in $O(ncf_1)$;

   (c) It applies the activation function ϕ to P^{12} in $O(nf_1)$, resulting in $P^{13} \in \mathbb{C}^{n\times f_1}$.

3. The second convolutional layer carries out similar operations with $c \to f_1$ and $f_1 \to f_2$, building $P^{23} \in \mathbb{C}^{n\times f_2}$ in $O(n^2f_1+nf_1f_2+nf_2) = O(nf_1(n+f_2))$.

4. Parts I and II of the unwind layer require a diversification based on the task to be solved:

   (a) In the link sign/existence/direction tasks, $O(m^{\text{train}}f_2+m^{\text{train}}f_2d) = O(m^{\text{train}}f_2d)$ due to 2 operations:

      i. Unwinding P^{23} into $U^{IL} \in \mathbb{R}^{m^{\text{train}}\times 4f_2}$ in (assuming random access) $O(m^{\text{train}}f_2)$.

      ii. Multiplying (linear layer) U^{IL} by $W^{IIL} \in \mathbb{R}^{4f_2\times d}$ to obtain $U^{IIL} \in \mathbb{R}^{m^{\text{train}}\times d}$ in $O(m^{\text{train}}f_2d)$.

   (b) In the node classification task, $O(nf_2+nf_2nf_2) = O(nf_2nf_2)$ due to 2 operations:

      i. Unwinding P^{23} into $U^{IN} \in \mathbb{R}^{n\times 2f_2}$ in $O(nf_2)$.

      ii. Applying a 1D convolution with a 0-dimensional kernel between U^{IN} and $C^{IIN} \in \mathbb{R}^{2f_2\times d}$, calculating $U^{IIN} \in \mathbb{R}^{n\times d}$ in $O(nf_2d)$.

5. The Softmax activation function requires linear time w.r.t. its input size, thus not playing any role in the analysis.