Master Thesis

Optimizing Bike Sharing System Flows using Graph Mining, Convolutional and Recurrent Neural Networks

Davor Ljubenkov (910418-3018)

davorl@kth.se

Academic Examiner: Šarūnas Girdzijauskas
Academic Supervisor: Amir Hossein Payberah
External Supervisors: Carlo Ratti, Fábio Duarte, Paolo Santi

Degree program: TIVNM - DASC

Subject department: EECS

Course code: II226X

EIT Digital Master School

July 8, 2019

Abstract

A *Bicycle-sharing system* (BSS) is a popular service scheme deployed in cities of different sizes around the world. Although docked bike systems are its most popular model used, it still experiences a number of weaknesses that could be optimized by investigating bike sharing network properties and evolution of obtained patterns.

Efficiently keeping bicycle-sharing system as balanced as possible is the main problem and thus, predicting or minimizing the manual transportation of bikes across the city is the prime objective in order to save logistic costs for operating companies.

The purpose of this thesis is two-fold; Firstly, it is to visualize bike flow using data exploration methods and statistical analysis to better understand mobility characteristics with respect to distance, duration, time of the day, spatial distribution, weather circumstances, and other attributes. Secondly, by obtaining flow visualizations, it is possible to focus on specific directed sub-graphs containing only those pairs of stations whose mutual flow difference is the most asymmetric. By doing so, we are able to use graph mining and machine learning techniques on these unbalanced stations.

Identification of spatial structures and their structural change can be captured using *Convolutional neural network* (CNN) that takes adjacency matrix snapshots of unbalanced sub-graphs. A generated structure from the previous method is then used in the *Long short-term memory artificial recurrent neural network* (RNN LSTM) in order to find and predict its dynamic patterns.

As a result, we are predicting bike flows for each node in the possible future sub-graph configuration, which in turn informs bicycle-sharing system owners in advance to plan accordingly. This combination of methods notifies them which prospective areas they should focus on more and how many bike relocation phases are to be expected. Methods are evaluated using *Cross validation* (CV), *Root mean square error* (RMSE) and *Mean average error* (MAE) metrics. Benefits are identified both for urban city planning and for bike sharing companies by saving time and minimizing their cost.

Keywords: Data Science, Data Visualization, Bike-Sharing Systems, Graph Mining, Time Series Prediction, Machine Learning, Deep Learning, Recurrent Neural networks, Convolutional Neural Networks, Shareable Cities, Urban Informatics

Referat

Lånecykel avser ett system för uthyrning eller utlåning av cyklar. Systemet används främst i större städer och bekostas huvudsakligen genom tecknande av ett abonnemang.

Effektivt hålla cykel andelssystem som balanseras som möjligt huvud problemand därmed förutsäga eller minimera manuell transport av cyklar över staden isthe främsta mål för att spara logistikkostnaderna för drift companies.

Syftet med denna avhandling är tvåfaldigt.

För det första är det att visualisera cykelflödet med hjälp av datautforskningsmetoder och statistisk analys för att bättre förstå rörlighetskarakteristika med avseende på avstånd, varaktighet, tid på dagen, rumsfördelning, väderförhållanden och andra attribut.

För det andra är det vid möjliga flödesvisualiseringar möjligt att fokusera på specifika riktade grafer som endast innehåller de par eller stationer vars ömsesidiga flödesskillnad är den mest asymmetriska.

Genom att göra det kan vi anvnda grafmining och maskininlärningsteknik på dessa obalanserade stationer, och använda konjunktionsnurala nätverk (CNN) som tar adjacency matrix snapshots eller obalanserade subgrafer.

En genererad struktur från den tidigare metoden används i det långa kortvariga minnet artificiella återkommande neurala nätverket (RNN LSTM) för att hitta och förutsäga dess dynamiska mönster.

Som ett resultat förutsäger vi cykelflden för varje nod i den eventuella framtida underkonfigurationen, vilket i sin tur informerar cykeldelningsägare om att planera i enlighet med detta.

Denna kombination av metoder meddelar dem vilka framtida områden som bör inriktas på mer och hur många cykelflyttningsfaser som kan förväntas.

Metoder utvärderas med hjälp av cross validation (CV), Root mean square error (RMSE) och Mean average error (MAE) metrics.

Fördelar identifieras både för stadsplanering och för cykeldelningsföretag genom att spara tid och minimera kostnaderna.

Acknowledgments

I would like to express a great appreciation to my thesis supervisor Amir H. Payberah for his patient and unconditional support I received throughout the thesis.

A special thanks to *Massachusetts Institute of Technology: Senseable City Lab* (MIT SCL) for providing me with a unique internship opportunity and giving many valuable remarks on my project, mostly by my external supervisors Carlo Ratti, Fábio Duarte, and Paolo Santi, but also all the amazing coworkers I was privileged to work alongside with.

Among those researchers, a big gratitude goes to MIT visiting professor and researcher Fábio Kon who helped me gain insight into the statistical analysis and data exploration part of the project.

My biggest support throughout the intense period of thesis writing was my family in Germany and Croatia: my parents, siblings and grandmother, who all unreservedly believed in me when deadlines seemed tough and inspiration was nowhere to be found.

Lastly, I would like to thank my extended family at *European Institute of Technology* (EIT), both fellow students and employees, as the last two years were the most amazing journey filled with entrepreneurial experiences and new friendships, which will continue to flourish even after acquiring our diplomas.

Contents

Abbreviations & Definitions

ACF = Auto-Correlation Function
ADAM = ADAptive Moment estimation
API = Application Programming Interface
ARIMA = Auto-Regressive Integrated Moving Average
BN = Batch Normalization
BPTT = Back-Propagation Through Time
BSS = Bike Sharing Scheme (Service)
CHS = Cycle Hire Scheme
CNN = Convolutional Neural Network
CNTK = Microsoft Cognitive Toolkit
CV = Cross Validation
D.C. = District of Columbia
DDGF = Data-Driven Graph Filter
DTW = Dynamic Time Warping
EDA = Exploratory Data Analysis
ELU = Exponential Linear Unit
FNN = Feedforward Neural Network
GPA = Generalized Procrustes Analysis
GCNN = Graph Convolutional Neural Network
GIS = Geographic Information System
GRASS = Geographic Resources Analysis Support System
GRU = Gated Recurrent Units
GUI = Graphical User Interface
HCA = Hierarchical Cluster Analysis
HITS = Hyperlink-Induced Topic Search
IP = Integer Programming
LCHS = London Cycle Hire Scheme
LDA = Latent Dirichlet Allocation
LSTM = Long Short-Term Memory
MAE = Mean Average Error
MAPE = Mean Absolute Percentage Error
MIT = Massachusetts Institute of Technology
ML = Machine Learning
MSE = Mean Absolute Error
NN = Neural Network
NP = Non-deterministic Polynomial
OD = Origin-Destination
OLS = Ordinary Least-Squares Regression
PCA = Principal Component Analysis
PIP = PIP Installs Packages
PLoS = Public Library of Science
RBM = Restricted Boltzmann Machine

ReLU = Rectified Linear Unit
RSS = Residual Sum of Squares
RF = Random Forest
RMSE = Root Mean Squared Error
RMSLE = Root Mean Squared Logarithmic Error
RNN = Recurrent Neural Network (not to be confused with Recursive Neural Networks)
RSS = Residual Sum of Squares
SCL = Senseable City Lab(oratory)
TF = TensorFlow
TfL = Transport for London
T-SNE = T-distributed Stochastic Neighbor Embedding
UDF = User Dissatisfaction Function
VGP = Vanishing Gradient Problem

List of Figures

List of Tables

1 Introduction

A *Bicycle-sharing system* (BSS) is a popular service scheme deployed in cities of different sizes around the world. It is a service in which bicycles are made available for shared use to individuals on a short term basis, for free or for a price. The user borrows and returns the bike by placing it in a "*dock*" and if the service doesn't use docks, then it is referred to as "*dockless*". Using these bike-sharing systems, people can rent a bike from one location and return it to a different or same place on need basis. They can rent a bike through membership (mostly regular users) or on demand basis (mostly casual users). This process is controlled by a network of automated stations across the city [1].

First BSS had its inception in 1965, when Amsterdam city councilman Luud Schimmelpennink proposed it as a way to reduce automobile traffic in the city center. After the city council rejected the proposal, Schimmelpennink's supporters distributed fifty donated white painted bikes for free usage around the town. The police, however, impounded the bikes, claiming that unlocked bikes incited theft [2].

In 1991, a second generation BSS was conceived in Denmark, offering a few hundred coin-operated bikes. In 1996, a third generation, now based on magnetic cards and several technological advances was initiated in England and continued to evolve within following years. But it was only when Lyon in 2005, and later Paris in 2007, made their wise deployments of several thousand shared bikes that these systems started to become known worldwide. A few years after that, similar programs spread throughout other continents and now, there are estimates that more than 18 million bikes are actively used in a variety of BSS worldwide.

An exponential growth has been observed in developed and developing countries, in large and small, dense and sprawling cities. One of the main arguments for the implementation of BSS is that they provide an effective alternative for the first-mile and last-mile problem, mainly when integrated with public transport [3, 4]. Data from the United States of America (USA) Department of Transportation's 2017 National Household Travel Survey (NHTS)[1] indicates that 35% of all car trips in the United States were shorter than 2 miles (3.218688 kilometers), and almost 50% or half of all car trips were less than 3 miles (4.828032 kilometers); a distance that could usually be covered with a reasonable amount of cycling. Thus, there are plenty of motivations and opportunities for the expansion of such systems both to new cities and within the cities that already have an existing basic BSS implementation. BSS have been assembled around the world in *ad hoc* manners - with little or no scientific, evidence-based planning. The complex dynamics of such systems and their interaction with the city life rhythm and other means of transportation is not yet fully understood. There are multiple business models, and public or private forms of funding BSS. Within the past few years, several BSS companies have gone bankrupt and most cities worldwide are still reluctant in con-

[1]https://nhts.ornl.gov/vehicle-trips

sidering bike sharing as an integral part of their mobility portfolio. However, with more data obtained, dynamics of such systems are slowly being investigated by scientists using research methods that inspect mobility flows, optimization algorithms and predictions.

The real expansion did not take place until the 21^{st} century when first municipal plans and larger scale business ventures, that offered service as we know of today, had been created. In general, cycling as a means of transportation in modern cities has grown significantly in the past ten years. The appearance of large scale bike-sharing systems and an improved cycling infrastructure are two of the factors that enabled this growth. An increase in non-motorized modes of transportation makes our cities more humane, decreases pollution, traffic, and improves the quality of life. In many cities around the world, urban planners and policymakers are viewing cycling as a sustainable way of improving urban mobility. Nevertheless, most cities still rely on 20^{th} century tools and methods for planning and policy-making. Recent technological advances enabled the collection and analysis of large amounts of data about urban mobility, which can serve as a solid basis for evidence based decision making.

The use of bicycles for short trips (defined as trips with distance below 5 kilometers) in medium to large cities for commuting, occasional, and leisure trips presents multiple proven benefits at the global, local, and personal level. In global terms, substituting motor vehicles with bicycles reduces carbon emission and energy consumption as well as negative environmental impact [5, 6]. With respect to local benefits to the city, an increase in the number of cycling trips in substitution of motorized trips helps mitigating traffic congestion, decreasing air and noise pollution, and the amount of required parking space [7]. In addition, it also brings several personal benefits for both mental and physical health [8]. Research shows that commuting to work on a bike also presents an advantage in relation to other active modes of transportation such as walking, since its higher cardio-respiratory intensity is associated with health benefits [9]. However, both pedestrians and cyclists are more exposed to accidents and injuries compared to a car or transit passengers [10]. In the case of cycling, the risk is aggravated when dedicated bike lanes are not available.

1.1 Problem

There are several problems that arise with such sharing systems.

Firstly, there is a fleet management problem. In order to keep BSS as balanced as possible, bikes are manually transported across the city at peak times. In priority areas, docking stations are continually replenished with bikes or the bikes being continuously removed from docking stations [11]. This is an expensive endeavour that BSS owners have to enforce in order for the system to run smoothly. These mathematical cost functions are hard to calculate and are currently being optimized by scientists in the operations research area. Their model focuses on the exact number of docks, in which each station is individually examined and later physically changed by adding or removing the docks based on the optimization simulation performed [12]. However, this method does not take the full-fledged prediction into account. More specifically, every day in the week is observed as a fixed property and is indistinguishable from the same day during any other week, month or year regardless of any external factors such as the weather, holidays, special events, seasonal differences etc. Also, optimization approach has a high computational cost because the problem is viewed as solvable in *Non-deterministic Polynomial* time (NP-hard), making it difficult to perform as fast as it is needed to be [13].

Secondly, in previous years the amount of data which was made available for researchers to work with was not sufficient enough, and in most cases the data time span covered was not exceeding more than a couple of months worth. In the best case scenarios, data analyzed had a time span of one year. Of course, this varies on the specific BSS whereabouts, but even the older BSS investigated were not explored to their full potential due to a small amount of data actually being publicly available. Another reason why researchers did not yet take an advantage of older systems could be that the infrastructure changed drastically and thus, first year (or two) worth of data is not representative anymore of the BSS as a whole.

Moreover, even though most of the visualization methods have already been covered in existing papers, there had been a lack of comparative studies with small to medium sized BSS that would try and investigate things such as: the underlying distribution laws of graph structures, prediction performances, visualization patterns or conclusions drawn about the mobility flow scenarios.

1.1.1 Knowledge Gap

Currently, there are not many state-of-the-art methods that use graph mining and machine learning in the area of BSS optimization of bike relocation strategies.

The ones that have been published in the last 2 years implemented Recurrent neural network (RNN) to predict station level demand [14] of New York's Citi Bike dataset and used Root mean square error (RMSE) as a validation metric [15]. However, the global overview of the network is not considered, and the method have never been used for small to medium-sized bike sharing networks but exclusively on the New York's dataset that had been already optimized using a specially tailored operations research algorithm devised at Cornell University [12].

Only one paper [16] that addressed Convolutional neural network (CNN) used Graph convolutional neural network with data-driven graph filter (GCNN-DDGF) model for station level hourly demand prediction in a large-scale bike-sharing network. Although this paper proposes quite an efficient method, it does not consider to use the method as a component in a comprehensive framework for dynamic bike rebalancing. Also, the model is not applicable to a directed graphs and it cannot learn a sparse graph filter.

Boston area BSS is a medium-sized BSS with a different amount of data and topology compared to larger ones. This specific size-wise property may incur a particular dynamics distinct from those investigated by the other papers, which mostly focus on New York dataset. Such papers might favor other methods than those that have performed well in related research for small and medium-sized BSS. Also, we would like to have computationally low-demanding method that takes little time to output the results and can efficiently use more data than any other paper as no other previous study analyzed more than two year worth of data.

Only Ai Yi et al.(2018) [17] addressed using Convolutional Long short-term memory (Conv-LSTM) but their work was focusing only on dockless bikes, weather is not taken into consideration and rebalancing was never discussed.

In conclusion, knowledge gap to be explored in this thesis includes a combination of improvements based on the research papers written during the last three years in the domain of bike sharing and rebalancing optimization:

- focus on the middle-sized docked BSS that have not been explored in detail before
- using secondary datasets such as the weather information
- combine mobility flows for both spatial and temporal patterns
- use a method that does not only predict, but also suggests how the bike rebalancing strategy should be utilized
- make a scalable rebalancing strategy so that it can be viewed not only from the entire system's point of view, but also from the municipality or neighbourhood level
- computational complexity is low and the utilized method does not require hours to calculate final result
- prediction accuracy is still high enough and comparable to other approaches
- research results provide a possible application for the Boston BSS, having a validated approximation of how many bike truck will be needed and which areas will most likely require a rebalancing process

1.1.2 Research Question

The improvements and additions on top of the shortcomings of proposed state-of-the-art models described previously will provide answers to the research question proposed by this thesis:

How weather information influences usage of middle-sized Boston BSS and can we get a reasonable prediction of bike usage using neural networks compared to relevant benchmarks?

The research question will be answered and evaluated in Chapter 7.

1.2 Purpose

The academic purpose of this work is to (1) explore specific properties of BSS through a statistical exploration analysis, and (2) to assess the relative strengths of different implementations of predictive models and their potential combination.

Analogously, the commercial purpose is to (1) obtain a model with powerful predictive capabilities, and to (2) reduce costs of bike relocation strategy by using an efficient label prediction, and (3) obtain a high-quality correctness score.

1.3 Goals

The goals of the work, in chronological order, is to:

- prepare and clean the Boston Blue Bike dataset
- find a suitable secondary dataset containing weather information, and use data wrangling methods to combine it with the primary data source
- use data exploration, statistical analysis and visualization to investigate bike sharing networks in order to get a better domain knowledge of BSS and problems that need to be addressed
- compare different Recurrent neural network prediction methods on the complete bike sharing dataset to find the best one to be used for data flow prediction where data flow is defined as the aggregated number of bike check-outs for each day
- define most unbalanced or asymmetric pairs of stations in the network for each month and create a subgraph containing this nodes stored as an origin-destination (OD) matrix
- use convolutional neural network to predict the label of the next subgraph that is most likely to emerge based on the preliminary data for the upcoming month
- utilize the chosen recurrent neural network on the output of the previous step to define the predicted flow and approximate the best strategy for the bike relocation in that specific configuration
- present the results by using appropriate validation metrics and conclude the thesis with some proposition and references for future work

1.4 Hypotheses

Prior to data exploration and uncovering relationships between BSS variables, it is necessary to gain some domain knowledge and use structured thinking about the problem. This form of problem inspection helps forming better features and eliminate possible biases. Some of the hypotheses that could influence bike demand:

Hypothesis 1:
Due to the hourly trend, a higher demand for bikes must exist during the rush hours. For example, late night period should have significantly lower demand compared to lunch hour.

Hypothesis 2:
On a daily trend scale, weekdays would need to have a much richer network compared to weekends or holidays.

Hypothesis 3:
Weather and season should highly influence bike demand numbers: rainy days, windy periods, higher humidity, and lower temperatures will probably have a positive correlation with bike demand. At least, this should be true in America and Europe. Things could be differently correlated in places with different climate like some Asian countries where correlation with humidity and temperature could be negative.

Hypothesis 4:
Some additional bike sharing influences could be city pollution levels or traffic congestion distribution.

Main Hypothesis:
However, main hypothesis to be examined in this thesis is the claim that we could use currently acquired bike sharing system data to predict future bike flows, especially for those stations that are considered to be problematic in a sense of their high relocation frequency and usage in general. Of course, this prediction is needed to perform within a certain degree of accuracy. Expected accuracy for the predicted dynamic flows should be around 90% and predicted spatial patterns around 70% when compared to the ground truth. Matching spatial patterns by using a trained 2-D CNN should perform reasonably well for highly similar items, but some fine-tuning could be needed as CNN might be the possible bottleneck when inspecting months with low bike flow usage numbers such as a winter period.

1.5 Contributions

Although a number of knowledge gaps, goals and hypotheses are mentioned, in this subsection only most important novelties are stated to be further discussed in the last chapter of this thesis.

Firstly, this includes an investigation of the secondary dataset containing weather information. Boston middle-sized BSS is a specific one because snow and rain seem to have no influence on the frequency of bike sharing usage. This is quite important as it implies that methods such as random forest are not as useful as they are for the bigger BSS.

Secondly, we want to propose a simple prediction method that does not underperform any of the investigated benchmarks. Even though no outperforming instances are observed, a comparison of the results is made and bottlenecks are identified.

1.6 Ethical Considerations

Due to the increasing pervasiveness of machine learning, it is crucial that there is a discussion about the safety, transparency and bias of machine learning systems. However, in the BSS the focus is mainly on rebalancing and even though information such as bike ID, gender and user type are available, it is not used to identify individuals. The bike sharing companies, however, may be storing user's personal information from the moment they register for the service, but the data is never publicly disclosed. From the company's perspective, in case they are using a more attribute-wise detailed data, they should be ensuring that a machine learning model does not leak its training data to an adversary that might be able to intercept a large number of queries, and in turn, see the statistics of the output distribution and cross-reference it to recover the original data. In addition to data driven ethical considerations, it is possible to infer from the data in case a bike had been stolen and in that case, a company should have regulations to investigate how and when to act if such events occur.

1.7 Sustainability

As argued in the introduction, using bike sharing services reduces carbon footprint and promotes healthy lifestyle. Optimization of environment friendly transportation will effectively attract new customers and users. Ultimately, the goal of this thesis is to work towards the overarching scope of creating a more sustainable and smarter cities, and many of related positive effects in the area of bike sharing networks is in accordance with "The 2030 Agenda for Sustainable Development Goals (SDG)", adopted by United Nations (UN)[2].

[2]https://sustainabledevelopment.un.org/

1.8 Limitations

The very first limitation is noticeable in the usage of exclusively docked BSS data. Originally, it was planned to have a slightly broader study were dockless bikes would be investigated as well, but that did not come to fruition as American and European companies that own such systems are not comfortable with sharing their data. It is important to mention that even in the case of a successful collaboration with such companies, the process of obtaining data involves a complicated legal procedure, and takes a couple of months in total which was not feasible regarding the time restrictions imposed upon the completion of this thesis in a timely manner.
Dockless data policies differ in China, but such approach was not taken into consideration due to a high volume of papers already written and specifically based upon this areas. Also, dockless systems or fourth generation bike systems, are much more popular in China compared to the rest of the world. There are over 30 private companies operating there, while dockless bikes are still in an experimental phase in both of the Americas and Europe.

Regarding the second limitation, it is not possible to produce the model with a perfect prediction accuracy or retrieve the highest precision of label assignment. This means that there will always be an error present depending on the volume of our data, implementation details of the specific model used, computation complexity and a variety of other variables, some of which are stochastic in their nature and therefore, out of our control. Still, establishing a performance baseline, defining model setups, and knowing our lower and upper bound is supposed to help us define our limitations empirically and mitigate any unwanted performances.

1.9 Thesis Outline

The following thesis report is organized as follows:

This first chapter introduced a short overview of BSS in general and its current development in the realm of flow predictions and rebalancing strategies using graph mining and machine learning. Knowledge gap had been defined and research question was stated, which will be answered in the following chapters.

The second chapter reviews some recent related work in the area of BSS by highlighting both strengths and shortcomings, as well as how the presented work is connected to the work contained in this very thesis.

Third chapter shortly summarizes the background and theory behind the machine learning methods used. A description of their architecture and list of the most important properties will be provided.

Fourth chapter starts of by presenting a case study of Boston area BSS and using data exploration and visualization techniques to summarize the most important properties and spatio-temporal changes to the network. Moreover, most unbalanced pairs of links between stations will be defined as a means to approximate an important metric to be used in the machine learning chapters to simplify the network only to those stations that are good candidates for rebalancing strategy.

Fifth chapter makes an overview of all the candidate methods for predicting dynamic patterns. A chosen method will be used for the stations identified in the next chapter.

Sixth chapter describes how to identify spatial structures and uses the unbalanced networks defined in Chapter 3. This is done by creating adjacency matrices of the past network configurations. Afterwards, adjacency matrices are labeled and used as an input for training the convolutional neural network which decides how might the newly observed spatial structure be labeled.

Seventh chapter discusses about how to apply the previously introduced methods in case of Boston BSS and describing the preliminary results. Here, our research question will be answered.

The last chapter closes the thesis with a conclusion and recommendations for future work based on the methodology presented throughout prior chapters.

2 Related Work

In this chapter, the most relevant recent work in the research area of BSS will be presented. Strengths, shortcomings and connections to this thesis will be briefly described. Each subsection contains those papers that fall under one specific BSS domain: spatial patterns, temporal patterns, dock optimization, visual analytics, comparative studies, mobility prediction.

2.1 Spatiotemporal Patterns

In the paper written by Grant McKenzie(2018) [18], docked and dockless BSS had been compared. Because of a sudden explosive growth in dockless bikesharing services, limited time was provided for municipal governments to set regulations and assess their impact on docked bikesharing programs. This was a motivation behind the paper to present an exploratory understanding of the differences in activity patterns between those two services. Results can be used to better inform urban planners, transportation engineers, and the general public. However, paper focuses exclusively on Washington, D.C. and most of the analysis is just exploratory, while results of the paper are preliminary due to lack of data. Comparisons were made between “Lime” (dockless) and “Capital” Bikeshare (docked). Lime is a private company and Capital Bikeshare is owned by the municipal government of D.C. (also Virginia and Maryland). Data analyzed included only a month of March in 2018 (238,936 individual trips). Temporal aspects were observed by calculating: mean duration, median duration, bike trip aggregation to the nearest hour of a week and independently normalized, with pattern subtraction. For spatial aspects, Voronoi tessellation was used to partition town map into polygons, by subtracting and intersecting these polygons with land use data from D.C.s Office of Planning. Regrading the network analysis, K-means algorithm was used for clustering the dockless locations with a number of clusters, and the conclusion made was that the existing docks are well situated. On top of that, Dijkstras algorithm for routing analysis was also implemented. In conclusion, suggestions mentioned that other modes of transportation should be taken into account, as well as behavioral motivation of users for selecting certain services.

2.2 Operations Research and Optimization of Docks

Daniel Freund et al.(2019) [12] uses a case study of “Motivate” (owned by Lyft, which also manages a vast number of US BSS such as Blue Bikes, Citi Bike, etc.) and collaborates with Cornell University in order to optimize the number of docks in New York. This is done from the point of view of operations research and uses optimization models. This is done with stochastic modeling, defining UDFs (User Dissatisfaction Functions) being a convey function, Poisson processes, M/M/1 queues, integer programming models, discrete gradient descent algorithm, and Kolmogorov’s backward equation. Optimization formulation is written like this:

$$\begin{aligned} minimize_{\vec{b}} \quad & \sum_i c_i(b_i, K_i) & & \\ s.t. \quad & \sum_i b_i & \leq B, & \\ \forall i \quad & 0 \leq b_i & \leq K_i. & \end{aligned} \tag{2.1}$$

where the number of docks at Station i is denoted by K_i, number of bikes by b_i, the UDF by $c_i(b_i, K_i)$, and the total number of bikes available by B. When the docks are being moved, K_i becomes the decision variable in addition to b_i in which case $\bar{K_i}$ is the number of docks at each station and we can write the constraint like:

$$\sum_i |K_i - \bar{K_i}| \leq 2k \tag{2.2}$$

In conclusion, this technique is very successful and already implemented in New York. However, the author himself admits that this method does not differentiate the temporal aspect which can affect bike stations when predicting the future bike tidal flows and does not take any secondary datasets into account.

2.3 Collaborative Visual Analytics

Beecham et al.(2014) [11] discuss automatic label classification of commuting behavior and inferring workplace of individuals in London cycle hire scheme (LCHS). Methods that are described include: weighted mean-centres, K-means clustering, kernel density estimation and community detection. They identify a fleet management problem and closed peak-time ,,loops" but do not attempt to solve it. Contributions are present in deriving customers' workplace areas and labelling commuting journeys based on a spatial analysis of travel behaviours. Data observed includes trips between 2011 and 2012, which makes a total of 5'048'000 journeys. Some new attributes have been created: e.g. distance from users home to the closest docking station, Recency frequency (RF) segmentation. Regarding the observation of spatio-temporal analysis, they used lines on a map (visual saliency) and fluctuations for each day of the week. Workplace centres for each cyclist have been derived by calculating: frequency of weighted centroids for docking station locations, using K-means clustering, hierarchical cluster analysis (HCA), and density estimation method [19].

2.4 Community Structures

Munoz-Mendez et al.(2018) [20] address a time-varying networks of bike stations and communities in London, where different motifs (loop, chain, star) and temporal evolution dynamics with extended time windows could potentially provide deeper insights into inherent relationships of spatially heterogeneous nodes (stations) or sub-networks (communities). They also suggest that instead of pure unsupervised learning, extended layers of urban systems should be used with an amenities to draw meaningful conclusions.

2.5 Comparing Cycling Patterns

Sarkar et al.(2015) [21] identified the problem of balancing between system usage and demand, which leads to a lack of available bicycles or free parking spaces at stations at various times of the day. Data used in this study had time span of 4.5 months, included 10 different cities, with a total of 996 stations and 108 samples. Focus of this paper was solely on the fullness of stations and not on mobility flows. Unsupervised learning was used to show the intrinsic similarities between the cities by utilizing predictability of stations occupancy and comparing cross-city error for each. What they found was that heterogeneity is observed only in bigger systems. Random forest and neural network were used to compare the accuracy of forecasting how many bicycles will be at a given station and at a given time.
Their paper also discusses how studies of shared bicycle systems have recently appeared in the data mining literature, and how Froehlich et al.(2009) [22] were the first to apply clustering techniques and forecasting models to identify patterns of behaviour in stations in Barcelona's ,,Bicing" system, explaining results according to stations location and time of day. A recurring conclusion across analyses is that spatiotemporal system usage patterns are tied to, and reflect, city-specific characteristics. By focusing on single city systems, these works seem to indicate that each city has a unique pattern, and that forecasting algorithms applied to each one may not be generalisable across the world.

O'Brien et al.(2014) [23] and Austwick et al.(2013) [24] characterize systems at the city level, comparing them in terms of system size (both by station count and geographic area), daily usage, and compactness; they build a hierarchy of cities that share similar characteristics and apply community detection algorithms to analyze similarities within systems.
In the paper examined, pairwise ground distances are computed between all locations recorded for a single station using the Haversine formula described by Robusto (1957) [25]. Aggregate occupancy time series are calculated with Pearson correlation used for comparison week-day and weekend. Hierarchical clustering with an agglomerative strategy (bottom-up approach) was used to identify which individual stations share similar behavioural traits across different cities. Selected metric to measure the similarity between station vectors was used and distance metric based on the dynamic time warping (DTW) algorithm. Finally, they mention a technique for finding the optimal alignment of two temporal sequences as a finishing step.

2.6 Mobility Prediction using Random Forest

Yang et al.(2016) [26] motivate their work by explaining that the primary issue for both users and operators is the uneven distribution of bicycles due to the demand and supply changing trends. This requires better bike re-balancing strategies which depend highly on bicycle flow modeling and prediction. Contribution of their work is two-fold: spatio-temporal bicycle mobility model based on historical data, and traffic prediction model mechanism per each station with sub-hour granularity. Relative error of an obtained prediction is used for an evaluation. The paper focuses on the city Hangzhou in China with around 2800 stations and 103 million records in a time span of one year. Methods used include: spatio-temporal modeling, estimating the number and time of check-ins at different stations, and using random forest theory to predict check outs given the time, weather, as well as real-time bike availability.

2.7 Mobility Prediction using Recurrent Neural Networks

Paper by Pan et al.(2019) [27] tackles bike sharing demand and supply by implementing a real-time predicting method, community detection, and a 2-layer LSTM RNN model for Citi Bike system in New York and Jersey City. In addition to the bike data, meteorology data is used as a secondary dataset. Training set includes year 2017, while test set consists of first three months in 2018. In total, 800 stations are identified. Regarding the evaluation, RMSE had been used. Motivation for the usage of deep LSTM is because it can handle a large amount of data in a reasonably small amount of time. One of the suggestions is to use these predictions in order to distribute the number bikes specifically to each station.

2.8 Predicting Station Level Demand using Recurrent Neural Networks

Again, in Chen et al.(2017) [14], bike shortage problem due to uneven bikes distribution is in the focus and efficient online balancing strategy is proposed as a solution. Unlike other papers where most of the research is about predicting global rental demand or rental demand at a cluster level, this paper considers station level demand prediction which could be more beneficial. Proposed architecture makes predictions for all stations at once. New York Citi Bike dataset is used with 8’081’216 individual trips. Regarding the methods, RNN is used on a station level for both rental and return, loss function uses Backpropagation through time (BPTT) and Vanishing gradient problem. Furthermore, data exploration is performed, correlation made between weather and number of rentals, and a baseline approaches are defined. Baseline approaches include: Ordinary least-squares regression (OLS), Random forest (RF) with 50 estimators, and Feedforward neural network (FNN) with 4 layers of Rectified linear unit (ReLU) activation function. Evaluation was made using RMSE and MAE.

3 Theoretical Background

In this section, an overview of the theory behind machine learning methods for identifying spatial structures and predicting dynamic patterns of bike sharing networks will be presented. Of course, the generalization is assumed for these architectures to be applied in any other domain or problem solving procedure other than the one presented in this thesis. A description of their architecture and a list of most important properties will be provided using references available online[3]. Lastly, a combination of disclosed methods will be outlined as a proposed pipeline for solving the specific research problem introduced in Chapter 1.

The following technologies, libraries and programming environments will be used to build the methods described in the consequent chapters.

Jupyter[4] is a web-based interactive computational environment for creating Jupyter notebook documents.
TensorFlow[5] (TF) is a Machine Learning framework for Python.
Keras[6] is a High-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano.
Scikit-learn[7] is a Machine learning library for the Python programming language.
Seaborn[8] is a Python data visualization library based on matplotlib.

3.1 Artificial Neural Networks

Artificial neural networks (ANN) are biologically inspired computer programs designed to simulate the way in which the human brain gathers its knowledge by detecting patterns and relationships in data and learns, trains through experience. An ANN is formed from hundreds of single units, artificial neurons or processing elements, connected with coefficients (weights), which constitute the neural structure and are organized in layers. ANN consist of three layers: input, hidden, and output. The power of neural computations comes from connecting neurons in a network. Each neuron has weighted inputs, transfer function and one output. The behavior of a neural network is determined by the transfer functions of its neurons, by the learning rule, and by the architecture itself [28]. The simplest ANN architecture is perceptron, but adding more layers is capable of solving more complex classification and regression tasks. One important parameter is activation function, which defines the output of that node given an input or set of inputs. There is a variety of activation functions to choose from:

[3]https://towardsdatascience.com/
[4]https://jupyter.org/
[5]https://www.tensorflow.org/
[6]https://keras.io/
[7]https://scikit-learn.org
[8]https://seaborn.pydata.org/

Sigmoid function (sigmoid) is mathematical function with a characteristic shaped sigmoid curve. Often, sigmoid function refers to the special case of the logistic function which generate a set of probability outputs between [0, 1] when fed with a set of inputs. The sigmoid activation function is widely used in binary classification.

$$Sigmoid(x) = \frac{1}{1 + e^{(-x)}} = \frac{e^{(x)}}{1 + e^{(x)}} \tag{3.1}$$

Hyperbolic tangent function (tanh) is an alternative function to the logistic sigmoid. Although function shape is similar, here output values can range between [-1, 1]. Thus, strongly negative inputs to the tanh will map to negative outputs.

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.2}$$

Softmax (softmax) activation function is used for multi-class classification. Softmax function calculates the probabilities distribution of the event over "n" different events. In general, this function will calculate the probabilities of each target class over all possible target classes. Later the calculated probabilities will be helpful for determining the target class for the given inputs.

$$Softmax(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} \qquad where \qquad i = 1, \ldots, J \tag{3.3}$$

Rectified linear units (ReLu) is an activation function, which is being used for hidden layers by the most recent artificial neural networks for the hidden layers. A rectified linear unit has output "0 "if the input is less than "0", and raw output otherwise. That is, if the input is greater than 0, the output is equal to the input.

$$ReLU(x) = max(0, x) \tag{3.4}$$

Leaky Rectified linear units (LeakyRelu) activation function works the same way as the ReLU activation function does, except that instead of replacing negative values of the inputs with "0", the latter get multiplied by a small alpha value in an attempt to avoid the outputs being the same value for any input.

$$LeakyReLU(x) = \begin{cases} x, & \text{if} \quad x \leq 1 \\ \alpha x, & \text{otherwise} \end{cases} \tag{3.5}$$

3.2 Recurrent Neural Networks

Recurrent neural network (RNN) is an ANN that has a neural connection pointing backward, so that it remembers its input due to internal memory. This makes RNN a perfect candidate for machine learning problems that involve sequential data. When RNN makes a decision, it takes into consideration the current input and also what it has learned from the inputs it received previously. A typical RNN has a short-term memory which consists of two inputs, the present and the recent past. It also uses *Backpropagation Through Time* (BPTT) in order to assign weights to the model, while training. However, it also suffers from exploding gradients when the gradient descent algorithm assigns a high importance to the weights it is not supposed to, and from vanishing gradients when the gradient values are so small that model stops learning. However, using *Long short-term memory* (LSTM) alleviates such problems.

3.3 Convolutional Neural Networks

Convolutional neural network (ConvNet or CNN) is a class of AAN that can process data which can be stored as a matrix, such as an image. It usually consists of three layers: a convolutional layer, pooling layer, and fully connected layer. So, this means that instead of using the normal activation functions defined for regular ANN, convolution and pooling functions are used as activation functions in CNN.

Convolution operates on two signals or two images. We have an input image, and a kernel, acting as a filter on the input image, producing an output image. Mathematically, a convolution of two functions f and g is defined as a dot product of the input function and a kernel function.

$$(f \circledast g)(i) = \sum_{j=1}^{m} g(j) \cdot f(i - j + \frac{m}{2}) \tag{3.6}$$

Pooling is a sample-based discretization process. The objective is to down-sample an input representation, reducing its dimensionality and allowing for assumptions to be made about features contained in the binned sub-regions.
There are 2 main types of pooling commonly known as max and min pooling. As the name suggests, max pooling is based on picking up the maximum value from the selected region and min pooling is based on picking up the minimum value from the selected region. Thus, CNN is a deep neural network which consists of hidden layers having convolution and pooling functions.

As an additional detail, before running the CNN, we can explore prepared image data using t-SNE and PCA algorithms.

T-Distributed Stochastic Neighbor Embedding is a non-linear technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. This algorithm attempts to find patterns in the data by identifying observed clusters based on similarity of data points with multiple features. However, after this process, the input features are no longer identifiable, and one cannot make any inference based only on the output of t-SNE. Hence, it is mainly a data exploration and visualization technique[29].

Principal Component Analysis or PCA is a linear feature extraction technique. It combines input features in a specific way that it is possible drop the least important feature while still retaining the most valuable parts of all of the features. As an added benefit, each of the new features or components created after PCA are all independent of one another[30].

3.4 Overview

Short overview of the method's pipeline used in this thesis is described here:

Firstly, raw data is obtained, prepared and cleaned, with secondary dataset (weather information) aggregated to it and made suitable for performing time-series analysis. Our benchmark is found in an Autoregressive moving average (ARIMA) method that uses basic data properties, such as mean and average, to set up the most rudimentary performance score defined as a lower bound to which we can compare every new method introduced. Simple recurrent neural network and deep recurrent neural network were investigated, but ultimately Long short-term memory (LSTM) is found to be the one producing the best prediction scores.

Then, based upon the data exploration discovery, most unbalanced links between station pairs are defined and transformed into adjacency matrices, also called ,,snapshots". Each adjacency matrix represents top ,,n" unbalanced links in a monthly manner.

Once again, we use data exploration for a first few days of the month for which we seek to predict an overall adjacency matrix of the most unbalanced links. First few days which correspond to the 1/3 of overall monthly flows are transformed into an adjaceny matrix that we use as an input for Convolutional neural network (CNN). Here, an assumption is made that first 1/3 of the monthly flows tend to converge to a specific monthly shape that CNN will guess based on all the previous monthly snapshots it is trying to learn from. As an output, CNN gives as a guess label that best describes the month (or rather its unbalanced spatial configuration) as a whole. Additionally, multivariate LSTM method can be used to introduce some novelty links to the labels predicted in order to score a higher prediction accuracy.

Finally, from the predicted adjacency matrix snapshot, it is clear which station pairs we have. For those stations, LSTM RNN will be utilized to predict two-way flows between station pairs and calculate the difference. The difference gained is an indication of an unbalance, and we can use the ground truth to rate the accuracy of prediction. The predicted differences are very valuable to bike sharing companies as they can plan how many additional bike truck to dispatch and where, during their bike relocation process.

4 Data Exploration & Statistical Analysis

Section 4.1 gives an overview of the input bike data, and Section 4.2 presents exploration and statistical analysis performed on that data. This section was investigated and analyzed in collaboration with Massachusetts Institute of Technology (MIT) visiting professor and researcher Fábio Kon[9] at Senseable City Lab (SCL). Section 4.3 focuses on visualizing bike mobility flows and formally defining the most unbalanced pairwise edges between the two stations.

4.1 Data Description

To illustrate the methodology of this case study, 7 years worth of data from the Boston BlueBikes[10] BSS was used. Bike sharing data was collected from the BlueBikes website, the largest Boston bike-sharing provider. Boston[11] is a relatively bike friendly city, having received a silver medal award[12] from the League of American Bicyclists in 2017. From 2007 to 2014, the bycicle lane mileage in Boston went from 0.03 miles (0.048 kilometers) to 92 miles (148.06 kilometers), with a decrease in bicycle accidents around 14% per year[31]. Boston's original bike-sharing system, Hubway, was launched in 2011 and it has been growing since then. In 2018, its name changed to BlueBikes and it now has over 1800 bicycles and 308 dock stations across Boston, Brookline, Cambridge, and Somerville. In the proposed analysis, nearly 8 million bike trips have investigated since the inception of the bike sharing program.

Below is a list of bike sharing data attributed with information about how they were represented:

- "*tripduration*": integer number with the unit measure in seconds, all trips longer than 24 hours (or 86400 seconds) were not taken into account because those trips are treated as faulty and not representative of the bike sharing system flow
- "*starttime*": exact time of the bike check-out with YYYY-MM-DD HH:MM:SS format representing the start of bike trip
- "*stoptime*": exact time of the bike check-in with YYYY-MM-DD HH:MM:SS format representing the end of bike trip
- "*start station id*": integer number representing the unique bike station identification code where the check-out of the bike occurred
- "*start station name*": string representing start bike station name

[9]https://www.ime.usp.br/ kon/

[10]https://www.bluebikes.com/system-data

[11]https://www.boston.gov/

[12]https://bikeleague.org/

- "*start station latitude*": float number representing geographic latitude of the start station
- "*start station longitude*": float number representing geographic longitude of the start station
- "*end station id*": integer number representing the unique bike station identification code where the check-in of the bike occurred
- "*end station name*": string representing end bike station name
- "*end station latitude*": float number representing geographic latitude of the end station
- "*end station longitude*": float number representing geographic longitude of the end station
- "*bikeid*": integer number representing a unique identification code for each bike vehicle
- "*usertype*": string, can be either "Subscriber" or "Customer" where the former is subscribed to use bikes for a longer period (monthly or annual) of time while the latter only pays for a one-time (single or a day pass) usage
- "*birth year*": integer number representing a year of birth of the particular user, in case they provided one
- "*gender*": Binary integer value, "0" for female and "1" for male, self reported by member

4.2 Case Study

Initially obtained descriptive statistics for Boston Blue Bikes data helps us understand usage patterns extracted from the data between 2011 and 2018. In Figure 1 - produced age, trip distance, duration, and speed histograms can be observed. Trip duration follows a log-normal distribution with a median of 10 minutes and with 75% of the trips taking under 16 minutes. On the other hand, the speed follows a Student's t-distribution, with men riding slightly faster than woman.

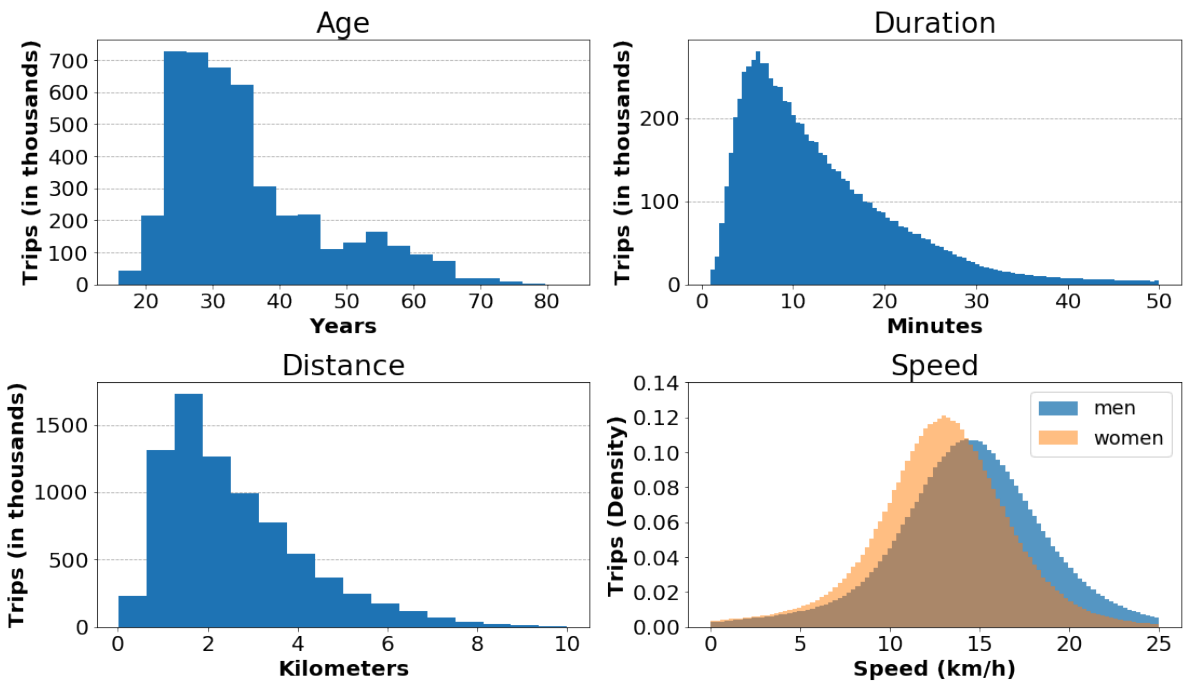


Figure 1: Descriptive statistics for Boston BlueBikes data

In Figure 2 we can see the evolution of the total number of trips per day for the entire bike sharing system. One can see both strong seasonal effects caused by the typical harsh winters in Boston, and the overall tendency for an increase in usage over the six years which is confirmed by the 12-month rolling average plotted. The men and women ratio shows not only that men use bike sharing more frequently but that the difference increases during the winter time. Finally, the figure also shows a slight increase in the proportion of female users in the past year. The cities of Boston, Cambridge, and Somerville have been improving the quality and extension of their cycling infrastructure. As women feel more comfortable and secure in the cycling tracks, the gap in usage for men decreases [32, 33]. However, it is still too soon to speculate if these will be a trend in the long run for the Boston area as well.

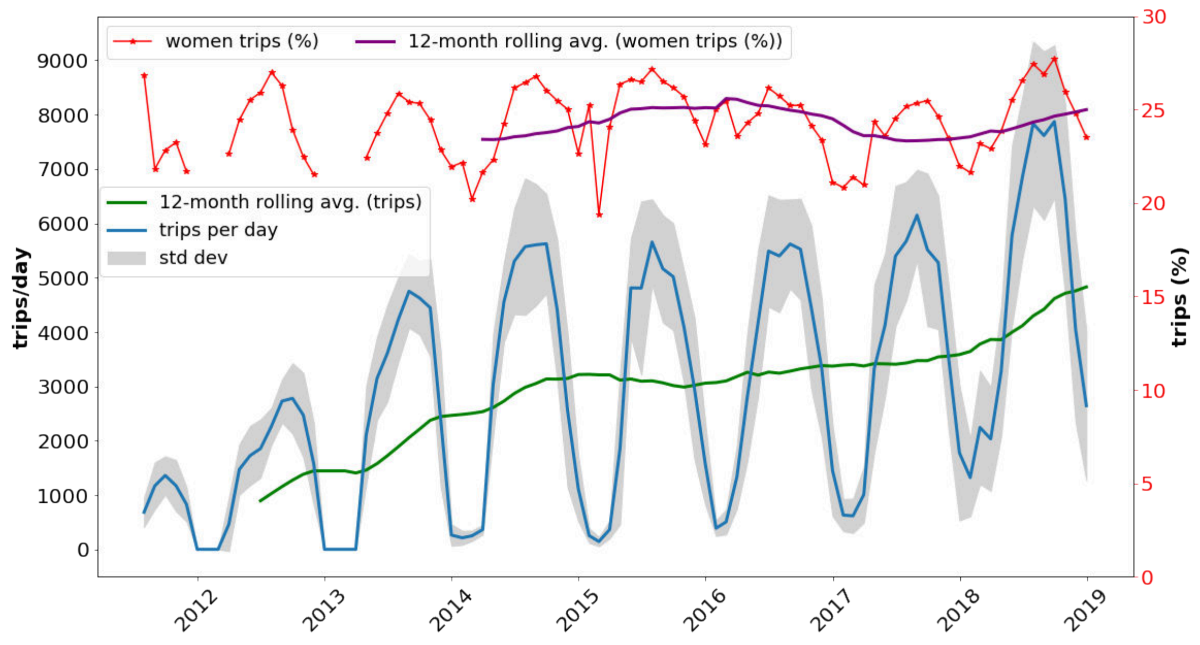


Figure 2: Evolution of trips from April 2013 to January 2019

For the analysis involving distances and speed, the road distance between two bicycle dock stations is estimated by using the GraphHopper[13] Application Programming Interface (API) over OpenStreetMap[14]; in particular, the bike mode route planner is used, which provides bike friendly routes. The bike routes suggested by the API are around 30% longer than the Euclidean distance, on average. Another option is to calculate distance based on the start and end point of longitude and latitude using haversine formula [34, 25]. The formula is given below:

$$a = \sin^2(\frac{\Delta\varphi}{2} + \cos\varphi_1 * \cos\varphi_2 * \sin^2(\frac{\delta\lambda}{2})) \tag{4.1}$$

$$c = 2 * a\tan 2(\sqrt{a}, \sqrt{(1-a)}) \tag{4.2}$$

$$d = R * c \tag{4.3}$$

where ϕ is latitude, λ is longitude, and R is Earth's radius.

Using the calculated speed, it is possible to detect the evidence of rider reckless behaviour. The most common reason for cycling accidents and fatalities is to get hit by a car [35]. Although car drivers are usually at fault for such accidents, according to the United States Department of Transportation, from 2010 to 2015, the most common

[13]https://www.graphhopper.com/
[14]https://www.openstreetmap.org/

bicyclist action prior to fatal accidents was the cyclists failure to yield right-of-way (in 34.9% of cases) [35]. A city government, then, may wish to develop an educational campaign to decrease the number of cyclists that ride bike dangerously fast. By analyzing the dataset and selecting the trips whose average speed was over 20 km/h, this analysis can be easily done. Given that the average speed of all trips is 13 km/h and that only 4.2% of the trips are above 20 km/h, we can consider that these fast trips have a large probability of being associated with cyclists riding dangerously fast. Profile of this speeders is as follows:

- 89% are men, while only 11% are women
- 50% of the speeders are between 21 and 32 years old, and although speeders are present in all ages under 52, the age range in which people have more tendency to drive dangerously fast is between 25 and 30
- the length of speedy trips is 20% longer than average and their duration is half that of an average of all trips
- a subscriber (usually, a resident) is 4.6 times more likely to be a speeder than just a customer (usually, a tourist)

4.3 Mobility Flows

Understanding where the major flows of cyclists are located within a city is the first step in providing urban planners with the knowledge required to draw a good mobility plan for urban cycling. Most previous work on BSS data analysis focuses on analyzing usage patterns of individual dock stations, without investigating the movements from one place to another, such as the origin-destination pairs of bike trips which can provide interesting insights on the punctual dynamics of the system [23, 36, 21, 37].

Because stations are normally distributed unevenly across the city, investigating each individual station does not provide an overall picture of city mobility dynamics for the urban planner. In one of the studies, Zhou (2015) [38] used a clustering algorithm to group together flows connecting dock stations in Chicago, identifying 378 relevant flows in the city for the year 2014. That was an interesting approach but showing so many flows to the user without any structure does not support policy making adequately. In addition, the computational complexity of the clustering algorithm might hinder the method's interactivity and fast usability.

For each trip, the location and time of origin - destination were used. Workdays present similar patterns among themselves but they differ greatly from weekends, so these classes can be treated separately. Within a single day, three different time periods are investigated: morning peak (from 7:00 to 10:00), lunch time (from 11:00 to 14:00) and afternoon peak (from 17:00 to 20:00) as their patterns differ significantly. Also, the average number of trips per hour in the dataset reduces significantly during the winter months.

An example of a morning peak distribution of bike check-outs by different neighbourhoods can be observed in Figure 3 and it is clearly visible that there is an excess of events happening in Boston Upper Back Bay, Boston North End, and Cambridge Harvard University area when observing aggregated mornings in the month of July 2018. This particular visualization was implemented using a Geographic Resources Analysis Support System for Geographic Information System (GRASS GIS)[15] open source software to define census tracts, and Leaflet JavaScript library[16] to allocate an amount of bike flows to a particular community in a presentable fashion. Census tracts are small, relatively permanent statistical subdivisions of a county or county within the United States, and they define the geographic borders between the communities.

Using heatmaps with different time granularities, as seen in Figure 4, these mobility flow changes can be observed even more clearly and in greater detail. Heatmaps were implemented using Folium library[17] on top of the Python ecosystem and Leaflet. Granularity can be changed to months, days, minutes, or even seconds.

[15]https://grass.osgeo.org/
[16]https://leafletjs.com/
[17]https://pypi.org/project/folium/

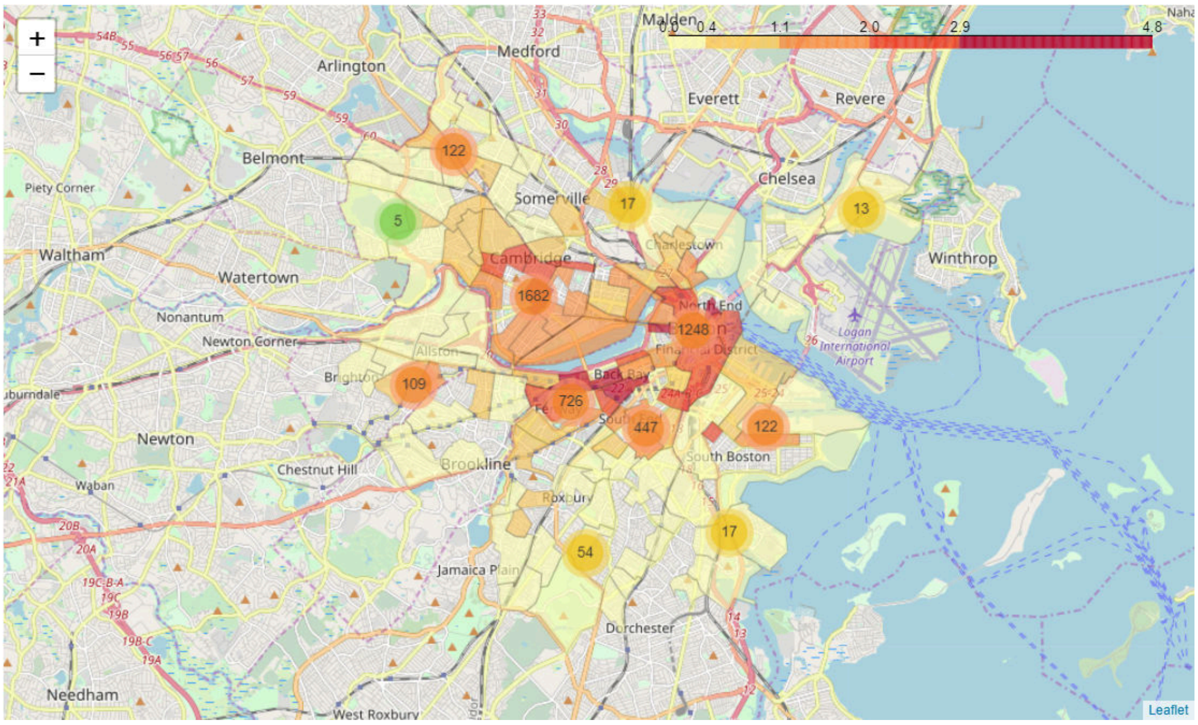


Figure 3: Morning trip check-outs clustered by neighbourhoods for July 2018

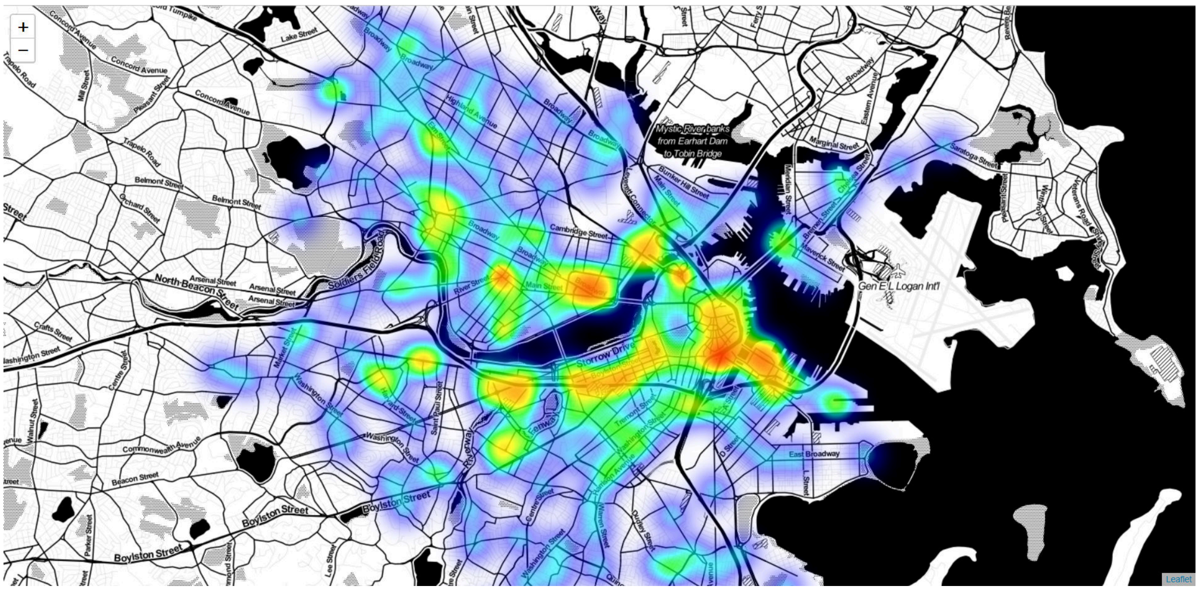


Figure 4: Morning trip check-outs heatmap for July 2018

An important visualization method is the one where mobility flows are represented as a directed graph, which is depicted in Figure 5. With the help of this method it is possible to define most pairwise asymmetric links between the nodes, which are of extreme importance in this thesis. Firstly, they contain the mobility flows most responsible for the unbalanced network which in turn, causes more frequent need for bike relocation. Secondly, these nodes will be used as a subgraph input for the Convolutional neural network (CNN) in Chapter 6 to gain an insight into future patterns that are most likely to be expected.

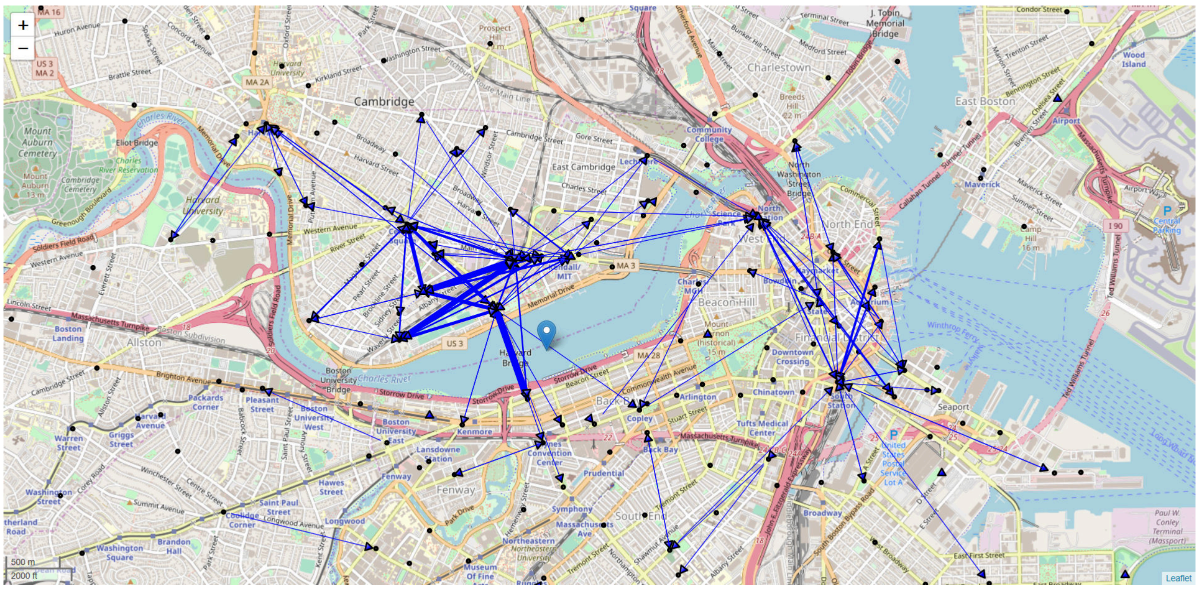


Figure 5: Mobility flows as a directed graph for July 2018

Algorithm 1 was used to find the most unbalanced links in the network whose absolute pairwise flow difference is larger than a critical number K, which is a positive integer. Two "for" loops are going through the origin-destination matrix of stations where flows are stored as an integer number. Every combination of station pairs is examined and for each one, coordinates are stored and flows are taken into account as the number of trips. In the "if" statement, we check if the difference between the station pairs is larger than K and if it is, we are drawing the bi-directional arrow to mark the unbalanced link.

```
for start id (odmatrix.row) do
    for end id (odmatrix.column) do
        start lat = stations.loc[start id,'lat'];
        start lat2 = stations.loc[end id,'lat'];
        start long = stations.loc[start id,'lon'];
        start long2 = stations.loc[end id,'lon'];
        end lat = stations.loc[end id,'lat'];
        end lat2 = stations.loc[start id,'lat'];
        end long = stations.loc[end id,'lon'];
        end long2 = stations.loc[start id,'lon'];
        num trips = odmatrix.loc[start id, end id];
        num trips2 = odmatrix.loc[end id, start id];
        if (abs(num trips - num trips2) > K) then
            draw arrow(start lat2, start long2, end lat2, end long2, num trips2);
            draw arrow2(start lat, start long, end lat, end long, num trips);
        end
    end
end
```

Algorithm 1: Most unbalanced links

This intuition that pairwise asymmetric links could be beneficial was confirmed by comparing the animated heatmap visualization of flow density (check-ins and check-outs) progression through different days and months, to the appearance of the most unbalanced pairs corresponding to the same time frame.
Moreover, using the most unbalanced pairs as an indication of bike shortage or abundance, it is possible to uncover some hidden problematic areas that are not so easily visible with the heatmap visualization. Also, having unbalanced pairs we clearly know where we should add or take bikes from when applying the relocation policy. This is something that would not be possible by just observing clusters during peak hours. Moreover, having subgraph of unbalanced pairs defined as an adjacency matrix data structure is an advantage when using it as an input for Convolutional neural network (CNN) in Chapter 6. In Figure 6 a representation of some 2018 unbalanced edges is plotted on the map for each month separately. It can already be seen that, although spatial configuration of these unbalanced subgraphs changes, some clear pattern rules can be concluded. For example, most of the unbalanced edges are concentrated in the Cambridge city center, but during warmer months subgraphs tend to expand to the very edges of the BSS network.

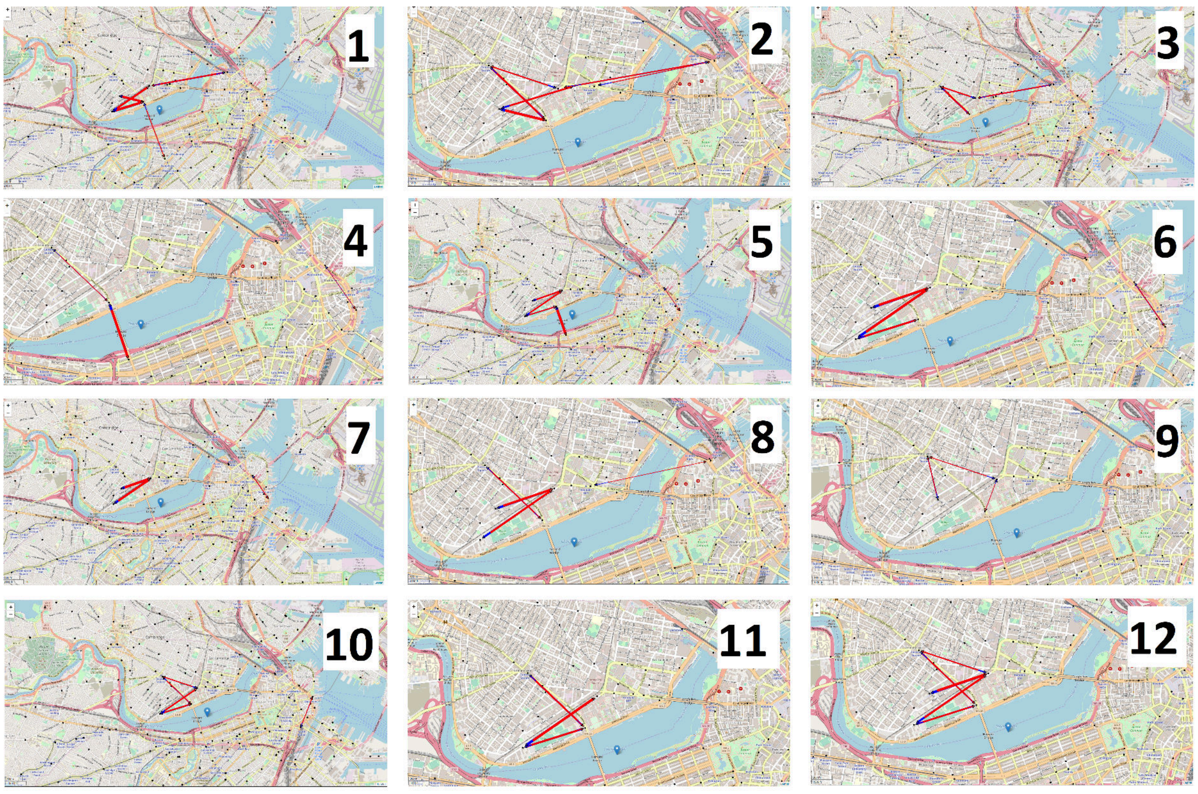


Figure 6: Most unbalanced links between stations for each month in 2018

5 Predicting Dynamic Patterns with RNN

In this chapter, a comparison between different predictive methods for time-series data will be examined. Primary dataset used is the aggregated number of bike check-outs for each day, and the secondary dataset contains weather details. Number of of bike check-outs, for reason of convenience, will be hereinafter referred to as “bike usage”. Best prediction method will be used on individual stations identified as the most unbalanced and obtained separately as an output of CNN in Chapter 6. The process of choosing the best prediction method will depend on the validation metrics retrieved and particular setting of parameters defined within the prediction model itself.

5.1 Bike Data Preparation

Primary dataset contains the BlueBike data in the time span between January of 2016 and March of 2018. In total, this accounts for 27 months of data worth, or 821 days. As mentioned in Chapter 3, all trips longer than 24 hours (or 86400 seconds) were not taken into account because those trips are treated as faulty and not representative of the bike sharing system flow. For the sake of simplification, only the number of bike check-outs (bike usage) is used and we will ignore most of other attributes for certain neural network demonstrations in this chapter. In addition, daily granularity is used which means that bike usage had been aggregated for each day independently based on the “starttime” being the exact time of bike check-out. This is denoted as variable “freq”, while we can also use “freqscaled” which had been normalized by dividing each bike usage value of each day with the highest bike usage observed (that exact value is 7405). This will create a range of values between zero and one, as for some neural networks it is sometimes easier to digest and process these normalized inputs. Moreover, one data entry was missing completely for March 13$^{\text{th}}$ in 2018, because a heavy blizzard happened on that day, and a zero value for bike usage was added manually to the dataset.

Speaking of normalizing inputs, when preparing data for artificial neural networks, a one-hot encoding should be enforced before fitting a machine learning model. This is because many machine learning algorithms cannot operate on label data directly and they require all input and output variables to be numeric. One-hot encoding can help by removing integer encoded variables and adding a new binary variable for each unique integer value. This method is applied for attributes such as: year, month, season, week-end, user type. Even though hot-encoding is explored, these additional attributes were not used in the final method as, ultimately, accuracy was not affected for predicting bike flows using a large amount of historic data. But, it could be argued that one-hot encoding and multiple attributes are crucial when dealing with smaller data and time spans.

5.2 Weather Data Preparation

Secondary dataset is a weather dataset previously acquired via "Kaggle"[18] website, but originally scraped from "Weather Underground"[19] platform. The dataset obtained did not have entries after the month of March in 2018, and that is why we took the same end period for the bike data. However, some of the original attributes were dropped out for the purpose of using it as an adequate input for recurrent neural network. For example, having an average temperature alongside high and low temperature seemed redundant, especially as it is so trivial to obtain average form the two latter mentioned extremes. Also, removing "Event" attribute is justified as we already have our information on snowfall and rainfall which is far more precise than just a boolean indicator of their presence. The produced weather dataset consists of: temperature (high and low), dew point (high and low), humidity (high and low), visibility (high and low), wind (high and average), high wind gust, snowfall, and precipitation.

Upon closer examination, it is important to notice that for the entry of, for example, 1.15 inch of snowfall it should be read as 1.15 foot which is 13.8 inch in total or around 35 cm. This conversion matches with the report for March 13th in 2018, when record snowfall height was measured in Boston[20]. All the snowfall measurements were changed accordingly.
Not only that, but certain days were missing from the dataset. As detailed inspection was performed for the year 2018, two days in March (21st and 22nd Of March 2018) were completely missing from the dataset and those rows needed to be somehow filled in or recreated.

5.3 Bike and Weather Data Aggregation

Now, having this weather dataset prepared, we would need to examine which of this attributes are the ones that correlate with the bike usage frequency the most. Simply by performing linear regression between "freqscaled" representing the bike usage and each one of the attributes, it is possible to empirically decide which attributes are more suitable to be kept as we don't want to burden machine learning algorithms with unnecessary attributes, thus slowing their accuracy and performance. Also, some of the more extreme dates with bike usage having an anomaly value were replaced with the mean of that month. This is because an extreme weather occurred (such as heavy snowfall) that is defined as a rare and impossible to predict event. Replacement with the monthly mean will result in a better performance of neural networks.

For each pair between bike usage and one of the thirteen attributes, an isolated scatter plot will be produced showing all the points representing a relation between the two

[18]https://www.kaggle.com/
[19]https://www.wunderground.com/
[20]https://www.boston.com/news/weather/2018/03/13/massachusetts-snow-accumulation-totals-march-13-2018

variables. In order to plot correctly, each of the attributes must be scaled by dividing their value with the maximum attribute value in existence. Then, a simple regression model will be applied and regression line can be observed on the plot. To evaluate which combination of bike usage and different attributes is the one with highest correlation (positive or negative), a coefficient of determination is defined and denoted as R squared. The value of R squared is typically taken as the percent of variation in one variable explained by the other variable, or the percent of variation shared between the two variables [39]. As a rule of thumb that correlation coefficient value between 0.7 and 1.0 are representing the strong linear relationship, and that means that only temperature attributes (high and low) can be used as relevant factors. In Figure 7 we can see that low temperature is positively correlated with the usage of bikes, while in Figure 8 we can see no correlation with high humidity. Table 1 contains all the correlation coefficients between different attributes and bike usage. The conclusion of using linear regression regarding the thesis project is that our final data aggregation should only combine temperature data with bike flows in order to minimize attributes used, while still retaining a satisfying significance on prediction possibilities.

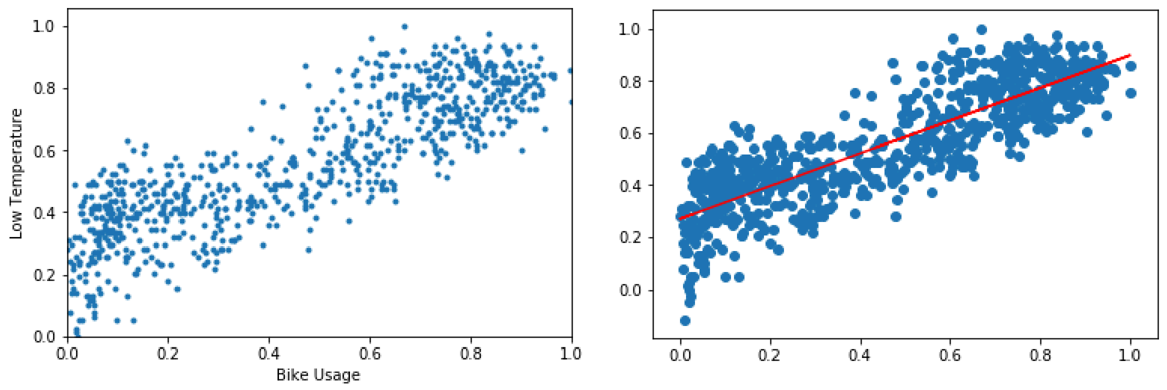


Figure 7: Correlation ($R^2 = 0.7$) between Low Temperature and Bike Usage

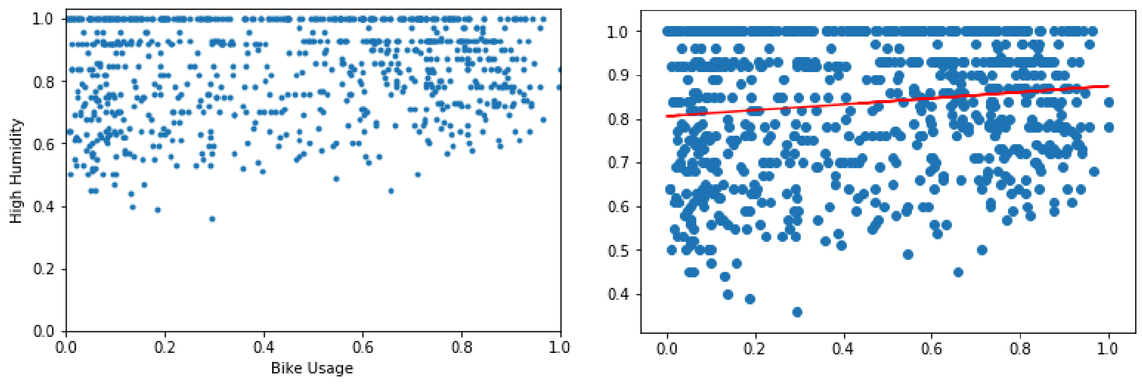


Figure 8: Correlation ($R^2 = 0.02$) between High Humidity and Bike Usage

Table 1: Correlation coefficient

Attribute	R^2
Low Temperature	0.7278257733186797
High Temperature	0.6914113376022786
Low Dew Point	0.6208915092809677
High Dew Point	0.5772408684832167
Low Humidity	0.0014386835458154446
High Humidity	0.018873022035322595
Low Visibility	0.043489802661249355
High Visibility	0.000004993336488734
Average Wind	0.07990729177445965
High Wind	0.07481014953823728
Wind Gust	0.07667909130885997
Snowfall	0.05167595979579198
Precipitation	0.014230885353750944

5.4 ARIMA

In a nutshell, bike usage or bike flow data represents a time series, which is a sequence of scalars that depend on time t. The objective of prediction is to guess future values by observing the past ones. Auto-regressive integrated moving average (ARIMA) is a generalization of an Autoregressive moving average (ARMA), and it is composed of two distinct models which explain the behaviour of a series from two different perspectives: Autoregressive (AR) model and the Moving average (MA) model. According to a number of sources [40, 41, 42] regarding univariate time series methods, when proposing new prediction methods, comparisons should be made against a naive and standard method such is an ARIMA model. This is to say that the model to be considered as a novel one should outperform the ARIMA model by comparing their performance metrics. An input for the ARIMA model will be bike usage time series in the form of the normalized number of bike check-outs ("freqscaled").

First step in implementing ARIMA is to test stationarity with an augmented Dickey-Fuller (ADF) test [43, 44] where we need to prove our reject our null-hypothesis:

- H_0 ... data is non-stationary
- H_1 ... data is stationary

A non-stationary time series show seasonal effects, trends, and other structures that depend on the time. Dickey-Fuller test in the case of data usage data produced a p-value of 0.371320 (Table 2). Because we got a p-value that is larger than 0.05, we proved the proposed null-hypothesis, which means our data has an unit root [45] and that the data can be used in ARIMA model once we verify rolling statistics.

Table 2: Augmented Dickey Fuller test

Test Statistic	**p-value**	**Lags**	**Observations**
-1.818492	0.371320	20	800

Rolling statistics indicates that summary statistics like the mean and variance do change over time, providing a drift in the concepts a model may try to capture [46]. In Figure 9 and Figure 10 we can observe and confirm these assumptions.

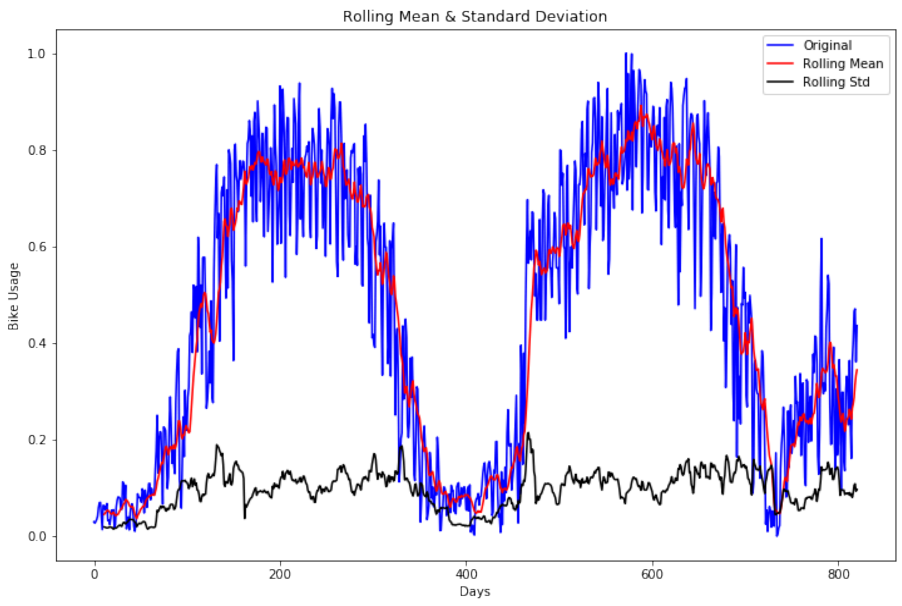


Figure 9: Rolling mean and standard deviation in ARIMA modelling

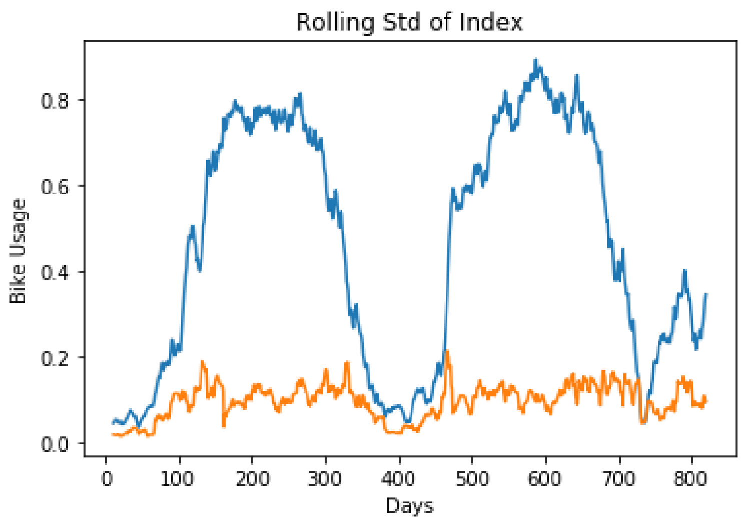


Figure 10: ARIMA model stationarity

As a last step before building an ARIMA model, we are going to observe auto-correlation function and partial auto-correlation function, which can be seen in 11.
Auto-correlation function (ACF) explains how well the present values of the series are related to its past values. We can see that for 30 lags there is a strong correlation above the value of 0.7. Lags are defined as observations with previous time steps and, the higher the lags, the further into the past we are trying to find correlation. Included in the 12 is the auto-correlation plot in case of an extreme case of choosing maximum number of lags (820). According to the literaure [47] this is exactly what we would expect: an ACF for the MA process to show a strong correlation with recent values up to the lag of "k", then a sharp decline to low or no correlation.
Partial auto-correlation function (Partial-ACF) represents the correlation of residuals, hence being a summary of the relationship between an observation in a time series with observations at prior time steps with the relationships of intervening observations removed[21]. Again, as described in theory [48], we would expect the plot to show a strong relationship to the first lag and then suddenly trailing off of correlation afterwards.

[21]https://machinelearningmastery.com/

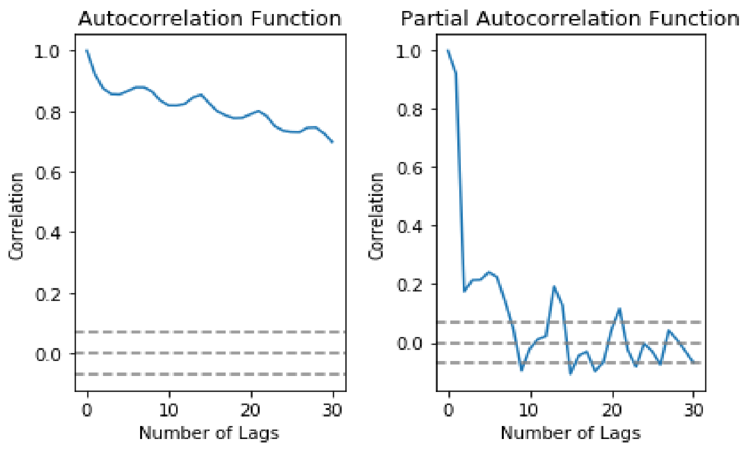


Figure 11: Autocorrelation functions for 30 lags

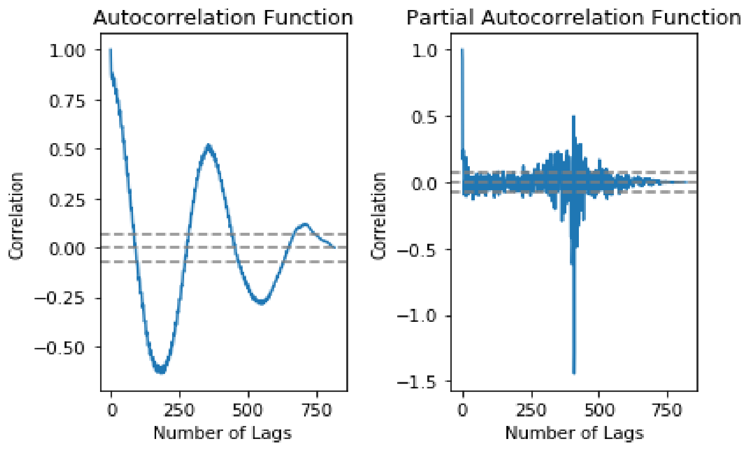


Figure 12: Autocorrelation functions for 820 lags

Finally, ARIMA model can be built based upon all the previous calculated necessities. Figure 13 shows the predicted bike usage in color red (with blue representing the actual bike flow), and also produces a number of error metrics. This metrics and performance results will be used as a benchmark for all the recurrent neural network predictions to be explored in this chapter.

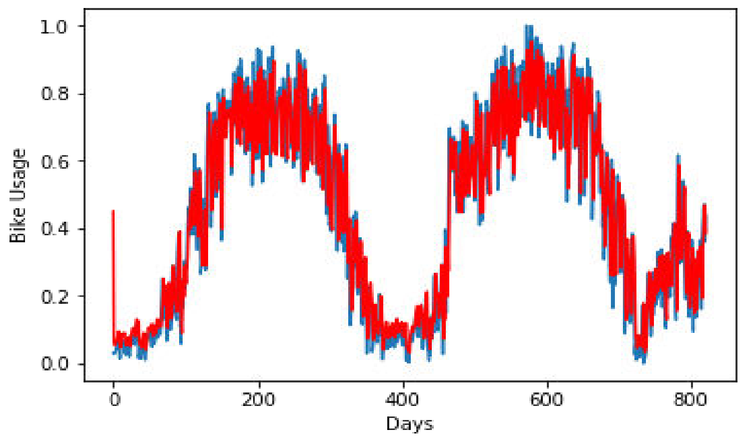


Figure 13: ARIMA predicted Bike Flow (2,0,0) coloured in red

The parameters of the ARIMA model (p,d,q) are defined as follows [49]:

- p: The number of lag observations included in the model, also called the lag order.
- d: The number of times that the raw observations are differenced, also called the degree of differencing.
- q: The size of the moving average window, also called the order of moving average.

In Table 3 and Table 4, evaluation metrics are summarized for ARIMA methods with two different settings for the number of lags: (2,p,q) and (20,p,q).

Table 3: ARIMA (2,0,0) evaluation metrics

Metric	**Score**
Elapsed time	60 miliseconds
RSS scaled	10.5776
MAE scaled	0.0856
MAE	632.1342
MSE scaled	0.0128
MSE	702543.2804
RMSE scaled	0.1135
RMSE	838.1785
Accuracy	0.8133

Table 4: ARIMA (20,0,0) evaluation metrics

Metric	**Score**
Elapsed time	161.28 seconds
RSS scaled	7.6769
MAE scaled	0.07269
MAE	536.7394
MSE scaled	0.00935
MSE	510006.1863
RMSE scaled	0.09669
RMSE	714.1471
Accuracy	0.8415

5.5 Simple RNN

Recurrent neural networks, also known as feedback neural networks, expand on the major shortcomings of traditional ANN. RNN are networks with loops, allowing information to persist and predict the future by observing the past. In a sense, RNN operates as a multiple feedforward neural network [50]. One big drawback of a simple RNN is that it has a vanishing gradient problem [51]. An input for the implemented simple RNN in this thesis consisted of three features: bike usage ("freq") and temperature ("High Temp (F)", "Low Temp (F)"). A desired output that we tried to predict was bike usage ("freq"). One hot encoding was implemented, but it was not necessary because the considered features were already strictly numerical. In Figure 14 we can see the simple RNN prediction of bike usage in color blue and the ground truth in color orange, for the test data. In Figure 15 training and validation loss functions are shown. Table 5 comprises of different simple RNN settings and results that are achieved when running them. Training dataset included first 500 entries, while the test set consisted of the last 321 entries.

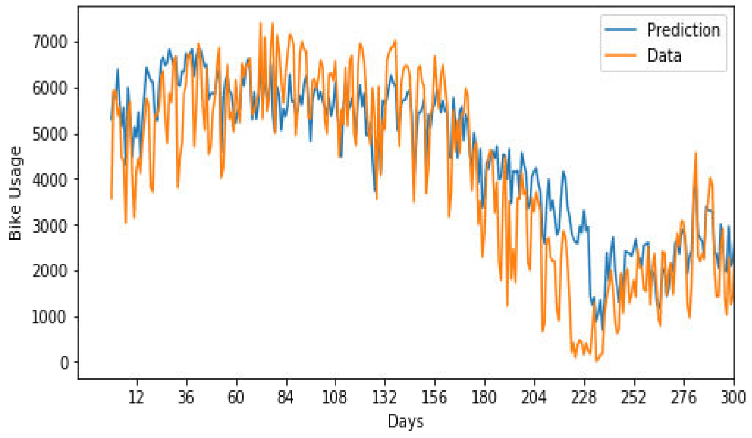


Figure 14: Predicted Bike Flow with simple RNN

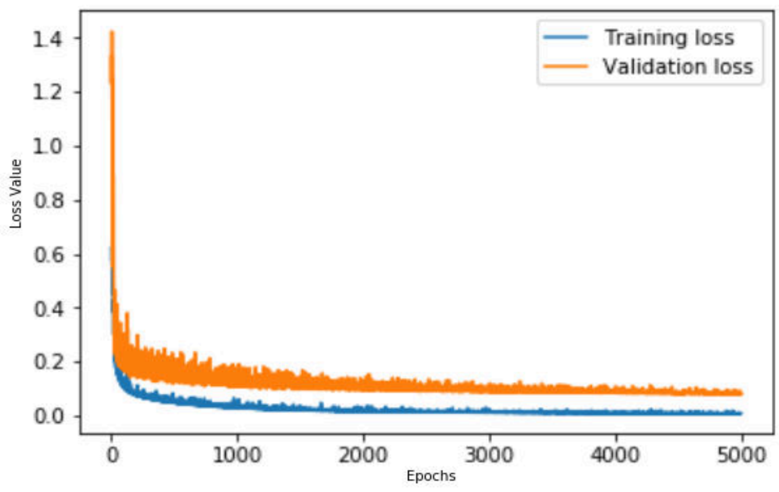


Figure 15: Simple RNN Loss Functions

Table 5: Simple RNN

Parameters & Scores	**Iteration 1**	**Iteration 2**	**Iteration 3**	**Iteration 4**
Test Set	321	321	321	321
Epochs	100	1’000	1’000	2’500
Learning Rate	0.001	0.001	0.001	0.001
Hidden Nodes	10	10	25	30
Elapsed time	0.4717 sec	5.0156 sec	5.5900 sec	14.2091 sec
MAE	1743.3761	1320.1722	1156.3751	783.8121
MSE	4269675.239	2612542.4243	1982988.6854	942435.7041
RMSE	2066.3192	1616.33611	1408.1863	970.7912
Accuracy	57.0039	67.4412	71.4808	80.6692

5.6 Deep RNN

In general, deep neural networks have multiple levels of hidden layers. This type of RNN benefits from the depth that these hidden layer levels create, and therefore outperforming the conventional, shallow RNN [52]. However, deep neural networks are often much harder to train than shallow neural networks [53]. All of this is true for deep recurrent neural networks as well. Also, concept of depth in RNN is not as clear as it is in feedforward neural networks [54]. Here, a performance of deep RNN was evaluated in the same manner as it was done for the simple RNN, analyzing the bike sharing dataset. Again, an input for the implemented deep RNN in this thesis consisted of three features: bike usage (“freq”) and temperature (“High Temp (F)”, “Low Temp (F)”), while a desired output to be predicted was bike usage (“freq”). In Figure 16, prediction with deep RNN can be seen, Figure 17 shows the prediction of the predefined hold-out data which is a form of cross validation method to ensure that predictions score is really consistent throughout the data, and Figure 18 represents the two loss functions and their values as the looping through epochs is running continuously. Finally, in Table 6, all the evaluation metrics and deep RNN settings are enlisted such as: activation functions, number of hidden nodes in each layer, number of epochs etc.

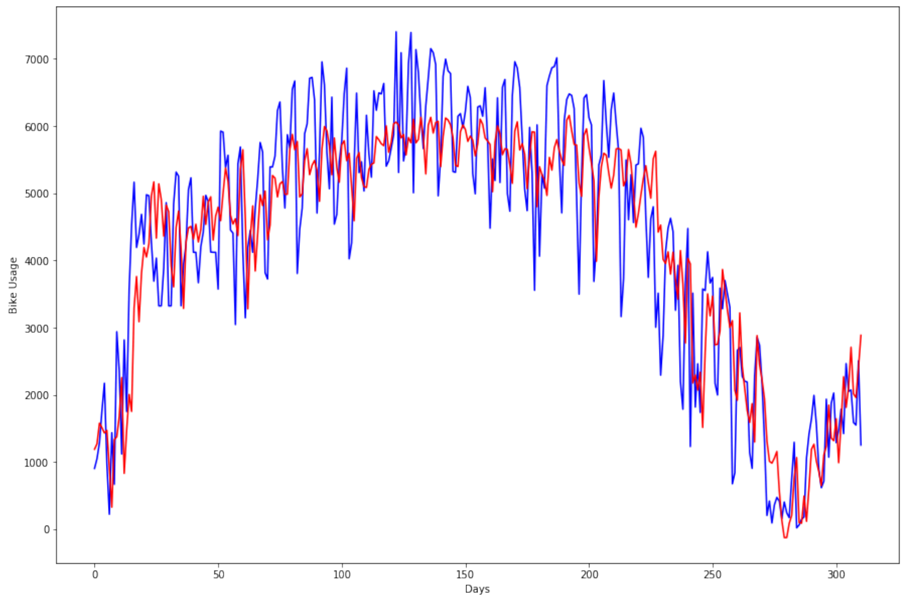


Figure 16: Deep RNN test datapredicted bike usage

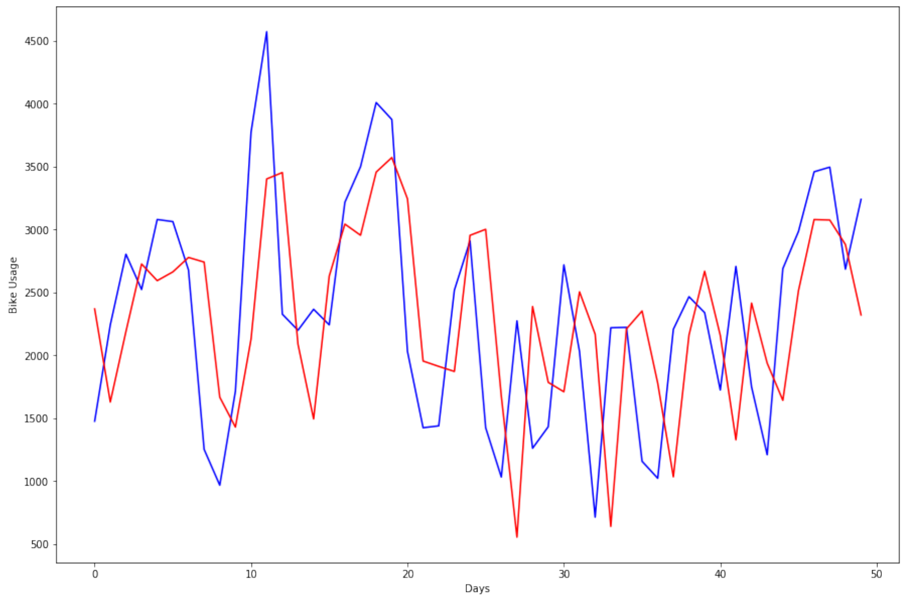


Figure 17: Deep RNN holdout data predicted bike usage (red) with 69% accuracy

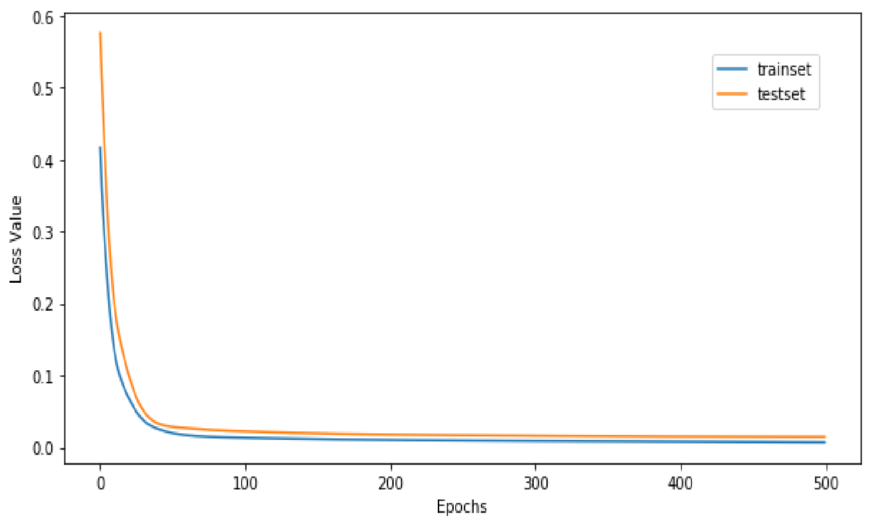


Figure 18: Deep RNN loss functions

Table 6: Deep RNN

Parameters & Scores	Iteration 1	Iteration 2	Iteration 3	Iteration 4
Test Set	321	321	321	321
Epochs	100	300	400	500
Learning Rate	0.001	0.001	0.001	0.001
Activation Functions	relu, relu	tanh, relu	tanh, relu	relu, tanh, relu
Hidden Nodes	4+2	8+8	16+8	24+12+8
Dense Layer	relu	relu	tanh	tanh
Optimizer	adam	adam	adam	adam
Elapsed Time	36.6 sec	160.87 sec	147.9 sec	249.7 sec
MAE	1558.2236	696.4746	694.7208	728.3816
MSE	3812000.29	626651.29	754513.30	854055.85
RMSE	1952.434	791.613	868.627	924.151
Accuracy	63.9155	83.8714	83.9120	83.1325

5.7 RNN LSTM

RNN Long short-term memory is a specific type of recurrent neural network capable of learning long-term dependencies. Not only does LSTM deal better with the vanishing gradient problem, but it is also well suited for making predictions based on a larger time series data [55]. In Figure 19, a prediction of scaled bike usage with LSTM can be observed, while in Table 7 performance metrics and LSTM settings are recorded. As an input, LSTM is considering only bike usage ("freq") feature.

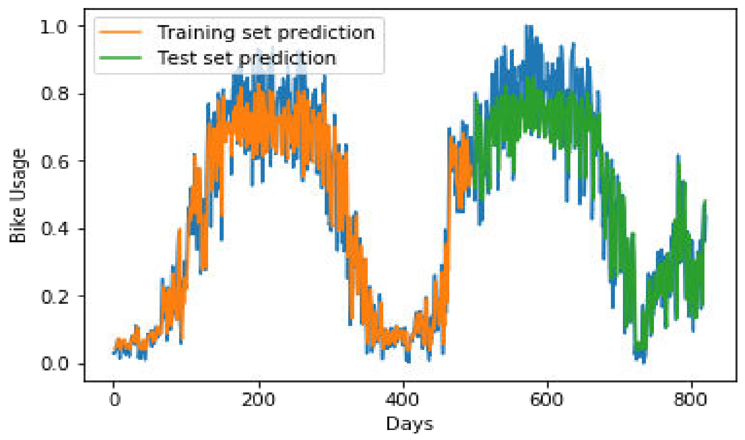


Figure 19: RNN LSTM predicted bike usage

Table 7: RNN LSTM

Parameters & Scores	Iteration 1	Iteration 2	Iteration 3	Iteration 4
Test Set	321	321	321	321
Epochs	300	300	250	300
Learning Rate	0.001	0.001	0.001	0.001
Activation functions	tanh, relu	relu, tanh	relu, tanh	relu, tanh
Hidden Nodes	16, 12	18, 12	20, 15	24, 18
Elapsed time	38.7 sec	60.6 sec	52.1 sec	67.6 sec
MAE	102.5375	92.4984	55.3264	96.7591
MSE	463712.94	405562.1	417788.88	421036.9
RMSE	680.9647	636.8375	646.3658	648.8735
Accuracy	0.88235	0.88368	0.88098	0.88201

As LSTM will be the method used for predicting dynamic patterns in this thesis due to its excellent performance, a cross validation of its results must be checked in order to make sure that LSTM prediction has reasonable values throughout the test dataset.

5.8 RNN Validation Metrics

Instead of a widely used K-fold cross validation metric, we would want to implement a series of test sets, each consisting of an equal number of observations. The reason behind this decision is because time series have a specific structure and K-fold method is not applicable. The corresponding training set consists only of observations that occurred prior to the observation that forms the test set [56]. This is supposed to ensure that no future observations are used in constructing the forecast. Also, two different approaches are used together in order to get a fixed training set with moving test set across different training set spans. This is just to ensure that accuracies obtained previously are valid.

As shown in the Figure 20, graphs in the first row have a training set of size 500, and a sliding test set of size 107. From left to right scores obtained by using parameters seen in Iteration 2 of Table 7 are: 91.4%, 87.6% and 82%, respectively. Second row has a training set of size 607 and a sliding test of the same size as the sliding test set mentioned before. Score for the test sets of the two graphs in the second row are: 90.7% and 81%, respectively. Now, it is easy to notice that as the sliding test window is moving further into the future, the prediction accuracy continues to drop down. But as long as we are trying to predict bike usage approximately 4 months into the future, the LSTM guarantees to produce at least 90% prediction score. Moreover, in Table 8 all of the methods are compared among themselves in a very concise way, using different evaluation metrics to clearly show how they all performed in predicting bike usage, given the training set of size 500, and a test set of size 321.

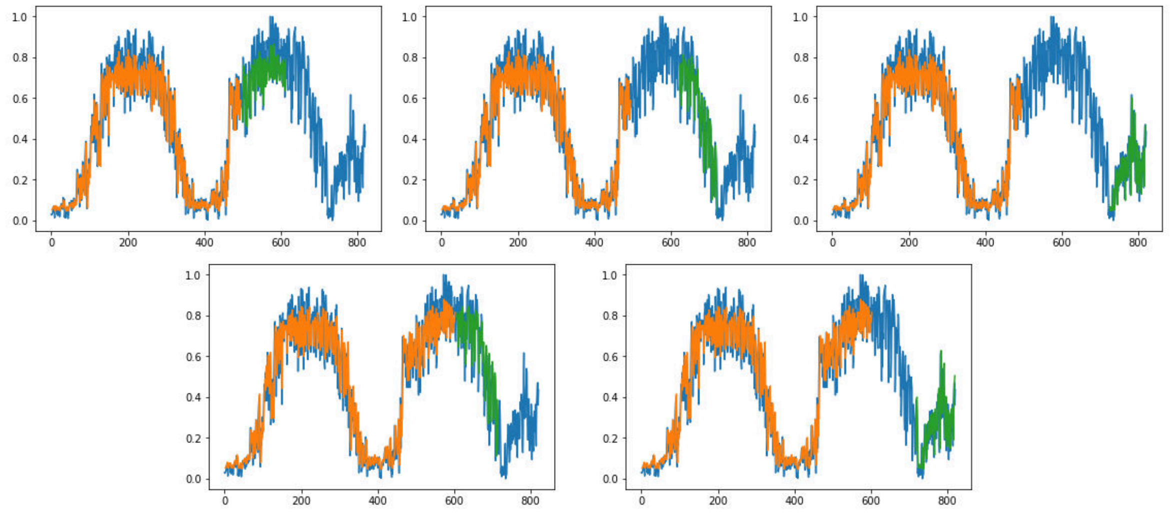


Figure 20: LSTM cross validation

Other evaluation metrics used in this chapter include Mean Absolute Error (MAE) and Root mean squared error (RMSE) which are two of the most common metrics used to measure accuracy for continuous variables. Mean absolute error measures the average magnitude of the errors in a set of predictions, without considering their direction, it is the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight. Root mean squared error is a quadratic scoring rule that also measures the average magnitude of the error. It is the square root of the average of squared differences between prediction and actual observation. Both MAE and RMSE are negatively-oriented scores, which means lower values are better. Taking the square root of the average squared errors has some interesting implications for RMSE. Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. This means the RMSE should be more useful when large errors are particularly undesirable [57].

$$MAE = \frac{1}{n}\sum_{j=1}^{n} |y_j - \hat{y_j}| \tag{5.1}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y_j})^2} \tag{5.2}$$

$$CV_k = \frac{1}{k}\sum_{j=1}^{k}\left(y_j - \hat{y_j}^{(-j)}\right)^2 \quad \forall \quad j = 1, \cdots, k \tag{5.3}$$

$$Accuracy = 1 - \frac{\frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y_j}|}{\frac{1}{n}\sum_{j=1}^{n}(y_j)} = 1 - \frac{MAE}{\mu(y_j)} \tag{5.4}$$

Using previously defined error metrics MAE and RMSE, we are now able to compare all of the available methods used for predicting dynamic patterns, as shown in the Table 8. Accuracy metric is defined as a MAE divided by the mean of the ground truth data values for the period observed. To get a percentage value we can subtract the calculate value from 1, and multiply by a hundred. Average accuracy is a mean value between all the different accuracies obtained within different iterations of the same method. Best accuracy is a value of accuracy for an iteration with a highest accuracy score. To conclude, the best ratio between runtime, accuracy and low error metrics is observed in LSTM RNN method, which was chosen as a one to be used for this project. And the RNN LSTM architecture setup to be utilizes is the one with the best accuracy score.

Table 8: Comparative analysis of explored prediction methods

Metric	**ARIMA**	**Simple RNN**	**Deep RNN**	**LSTM RNN**
Average Time	80.67 sec	6.3216 sec	148.7675 sec	54.75 sec
Average MAE	584.4368	1250.933875	919.45015	86.78035
Average RMSE	776.1628	1515.408203	1134.20625	653.260375
Average Accuracy	82.74%	69.148775%	78.70785%	88.2255%
Best Accuracy	84.15%	80.6692%	83.9120%	88.368%

6 Identifying Spatial Structures with CNN

In this section, pairwise most asymmetric edges between nodes representing most unbalanced flows between stations, which had been previously retrieved in the Chapter 4, are transformed into adjacency matrices and prepared as an input for convolutional neural network. Before running the CNN model, training set of matrices needs to be labeled because the goal of this method is to make sure that CNN correctly classifies test matrices to a correct label. In case of a false classification, we still want to maintain those errors to be spatially close to the expected stations, thus making mistakes as minimal as possible.

CNN is a neural network, a regularized version of multilayered perceptrons, and mostly applied in image recognition and classification areas.CNN architectures are usually built with the following layers: convolution layer, rectified linear units (ReLU) layer, pooling layer, fully connected layer and loss layer [58]. Because bike flows and, in particular, unbalanced stations tend to form a specific graph configuration which changes over time, in Chapter 4 that notion was used to visualize the changing shapes. This phenomenon was the main motivation to try and figure out how could these observed spatial configurations be recognized and classified into the predefined set of pattern labels with the help of CNN.

6.1 CNN Data Preparation

The first step is to take so called data “snapshots” of the directed subgraph that is formed by the most unbalanced stations and their links, already defined in Chapter 4. These snapshots have a certain time granularity, so we can observe different unbalanced graphs for each year, month, week, or day. Suppose that the granularity is monthly, which means that during one year worth of time, we will have 12 unbalanced graphs. In the case discussed here, we will take years 2017 and 2018 into consideration amounting to a total of 24 distinct unbalanced graphs when considering monthly granularity. So, this snapshots

Using the data exploration findings and Algorithm 1 from Chapter 4, we can observe in Table 9 and Table 10 monthly snapshots of the top three most unbalanced pairs of stations for each month in years 2017 and 2018. “Gap” columns represent the exact unbalanced flow value between the two station, which is calculated as a flow difference between the pair observed.“Station” columns inform us which station pair in question is sharing the unbalanced flow.

Table 9: 2017 most unbalanced station pairs

Time	**Gap1**	**Stations1**	**Gap2**	**Stations2**	**Gap3**	**Stations3**
Jan 2017	64	Ames,Vassar	46	Broadway,Post	41	Central,Mass
Feb 2017	72	Vassar,Stata	60	Vassar,Ames	45	Vassar,Mass
Mar 2017	68	Vassar,Stata	54	Vassar,Ames	53	Stata,Cambridge
Apr 2017	121	Vassar,Stata	75	Stata,Pacific	68	Vassar,Ames
May 2017	187	Stata,Vassar	108	Stata,Pacific	98	Nashua,South
Jun 2017	174	Vassar,Stata	118	Stata,Pacific	88	Nashua,Rowes
Jul 2017	162	Vassar,Stata	113	Davis,Teele	110	Mass,Boylston
Aug 2017	115	Davis,Teele	103	South,Nashua	97	Stata,Vassar
Sep 2017	140	Pacific,Stata	137	Mass,Beacon	114	Nashua,South
Oct 2017	147	Stata,Vassar	131	Stata,Sidney	102	South,Nashua
Nov 2017	114	Davis,Teele	88	Beacon,Mass	68	Mass,Central
Dec 2017	81	Mass,Pacific	53	Stata,Mass	49	Central,Stata

Table 10: 2018 most unbalanced station pairs

Time	**Gap1**	**Stations1**	**Gap2**	**Stations2**	**Gap3**	**Stations3**
Jan 2018	51	Nashua,Stata	49	Stata,Vassar	46	Mass,Pacific
Feb 2018	77	Central,Mass	62	Nashua,Stata	56	Mass,Pacific
Mar 2018	80	Nashua,Stata	71	Central,Stata	69	Central,Mass
Apr 2018	96	Mass,Central	94	Mass,Beacon	72	Rowes,Cross
May 2018	148	Stata,Vassar	99	Rowes,Cross	96	Stata,Pacific
Jun 2018	161	Stata,Vassar	115	Rowes,Cross	115	Stata,Pacific
Jul 2018	172	Stata,Pacific	145	Stata,Vassar	132	Rowes,Cross
Aug 2018	198	Stata,Vassar	195	Stata,Pacific	140	Central,Mass
Sep 2018	164	Stata,Central	109	Central,Pacific	97	Stata,Mass
Oct 2018	151	Central,Stata	129	Mass,Vassar	123	Central,Mass
Nov 2018	177	Stata,Vassar	130	Mas,Vassar	103	Mass,Central
Dec 2018	125	Stata,Vassar	108	Mass,Central	106	Stata,Pacific

Now, it is necessary to convert obtained station names into some form of identification numbers. If we aggregate both years together and find the exact number of total distinct stations, it becomes clear that there are a total of 20 such identification numbers we would need to allocate to each one of the station. Before the best identification number placement strategy can be discussed, in the Image 21 we can observe most unbalanced stations and network edges for year 2017 and in Image 22 we can see the same for year 2018.

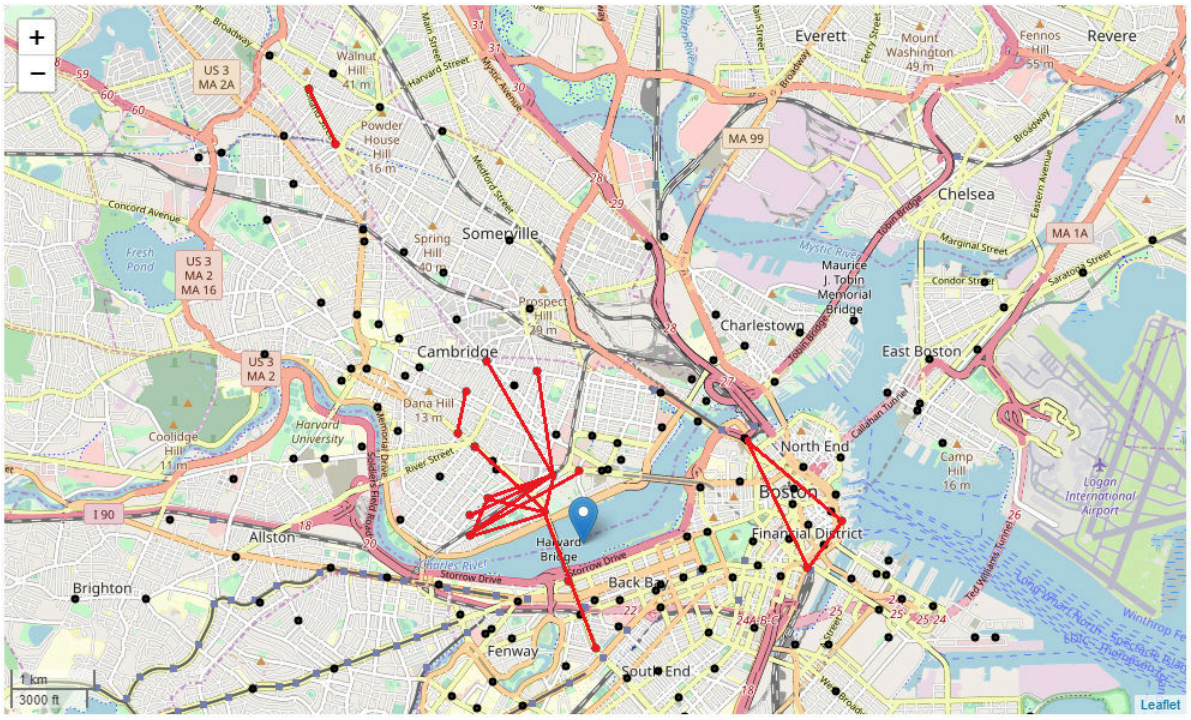


Figure 21: Aggregated unbalanced edges for year 2017

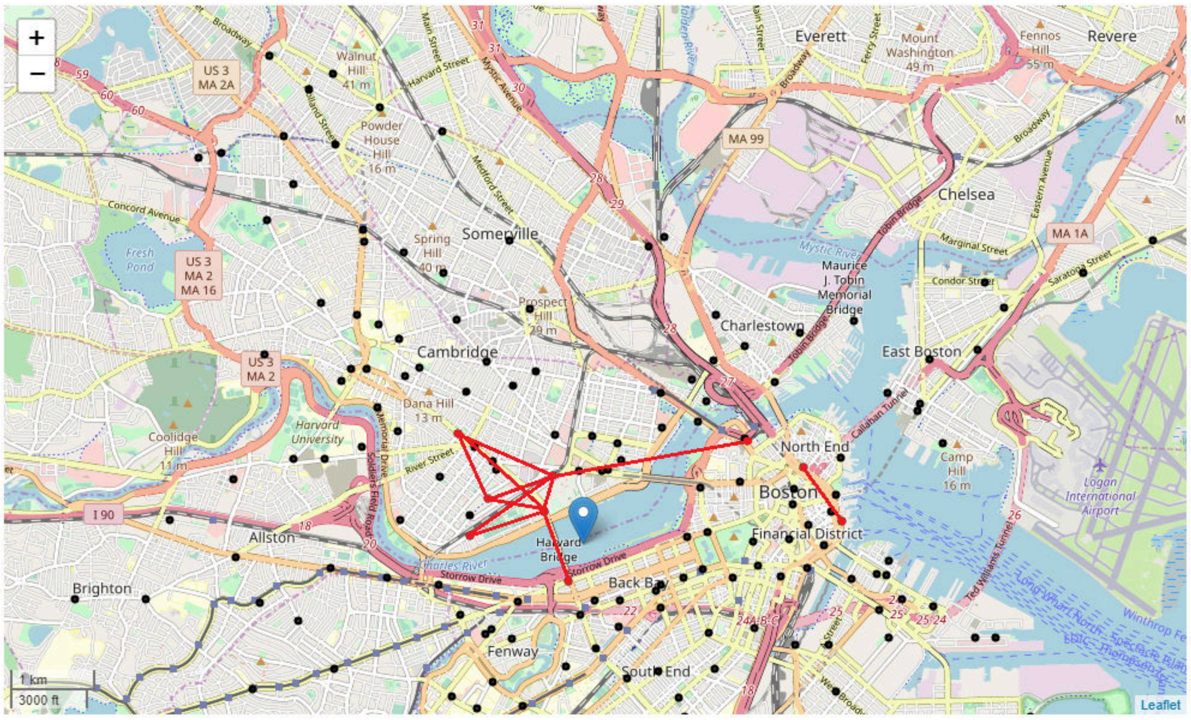


Figure 22: Aggregated unbalanced edges for year 2018

Allocation of identification numbers is done in the manner that mimics the way these stations are spatially placed within the area of Somerville, Cambridge and Boston. For example, stations Teele and Davis in Somerville are given ID numbers 1 and 2 because they are located in the far north-west part of the map as seen in Image 23. As we get closer to the Cambridge city center, other stations are given their unique ID numbers. Last couple of numbers are assigned to the stations in Boston area. Now, Table 9 and Table 10 are translated into Table 12, where each station is represented by its distinct ID number (shown in Table 11). Transforming strings into identification integers is an approach that had been done in order to utilize the next step, where each obtained snapshot will be represented in a form of a square, symmetric and hollow matrix.

Stations were assigned their identification numbers using a simple heuristic method. This method used some prior domain knowledge and is not an automated assignment process. Assignment started in the most north-west station and continued to the south-east while trying to group stations from the same neighbourhoods together, which makes some assignments to spiral in space. This could, of course, cause some slight problems in case we have a larger number of stations as it would be really hard to identify the most optimal way to assign the numbers. Moreover, this way of representing station is sensitive to permutation operations as it could lead to a greater number of errors in those cases where stations from completely different areas are mixed with each other.

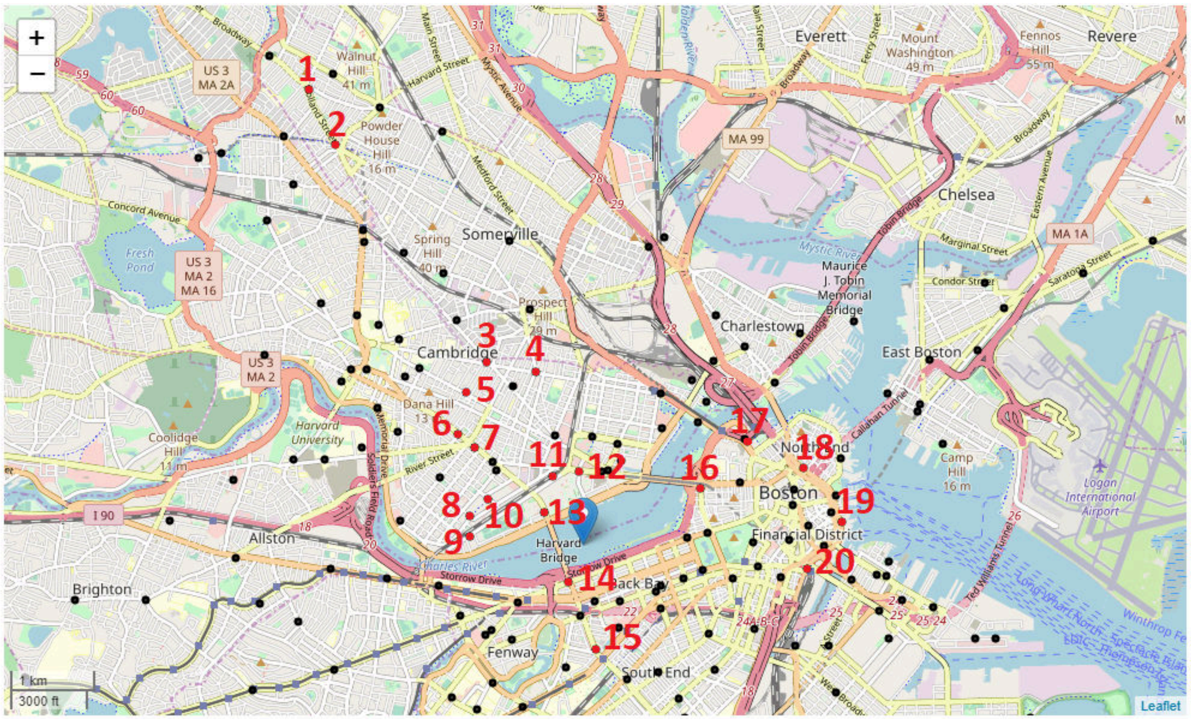


Figure 23: Appearances of all unbalanced stations during both 2017 and 2018

Table 11: Encoding the stations

Station name	**Identification Number**
Teele	1
Davis	2
Inman	3
Columbia	4
359 Broadway	5
Post Office at Central	6
Central	7
Sidney Research	8
Vassar	9
Pacific	10
Stata	11
Ames	12
Mass	13
Beacon	14
Boylston	15
Charles	16
Nashua	17
Cross	18
Rowes	19
South Station	20

Table 12: Encoded most unbalanced station pairs in 2017 and 2018

Time	**Top1**	**Top2**	**Top3**
Jan 2017	(9,12)	(5,6)	(7,13)
Feb 2017	(9,11)	(9,12)	(9,13)
Mar 2017	(9,11)	(9,12)	(11,4)
Apr 2017	(9,11)	(11,10)	(9,12)
May 2017	(11,9)	(11,10)	(11,12)
Jun 2017	(9,11)	(11,10)	(11,12)
Jul 2017	(9,11)	(1,2)	(13,15)
Aug 2017	(2,1)	(20,17)	(11,9)
Sep 2017	(10,11)	(13,14)	(17,20)
Oct 2017	(11,9)	(11,8)	(20,17)
Nov 2017	(2,1)	(14,13)	(13,7)
Dec 2017	(13,10)	(11,13)	(7,11)
Jan 2018	(17,11)	(11,9)	(13,10)
Feb 2018	(7,13)	(17,11)	(13,10)
Mar 2018	(17,11)	(7,11)	(7,13)
Apr 2018	(13,7)	(13,14)	(18,19)
May 2018	(11,9)	(18,19)	(11,10)
Jun 2018	(11,9)	(18,19)	(11,10)
Jul 2018	(11,10)	(11,9)	(18,19)
Aug 2018	(11,9)	(11,10)	(7,13)
Sep 2018	(11,7)	(7,10)	(11,13)
Oct 2018	(7,11)	(13,9)	(7,13)
Nov 2018	(11,9)	(13,9)	(13,7)
Dec 2018	(11,9)	(13,7)	(11,10)

In order to be able to use CNN, it is necessary to transform these unbalanced graphs into 24 adjacency matrices consisting of zeroes and ones, where each value “1” indicates that there is a directed edge from the station represented as matrix row with index “i” to the station denoted as a matrix column with index “j”, for that specific (i,j) tuple. Of course, because we are dealing with the bidirectional unbalanced graphs, adjacency matrices will be symmetric and also hollow meaning that we can observe only zero values on the main diagonal, thus avoiding nodes with self-loops. Now, these sparse matrices can be simply imagined as pixelated images where “1” indicates a pixel existence, and it is something that can be send as an input to CNN. However, there is a crucial step in transforming graphs into matrices and, as explained before, that step is to be aware of the fact that relative coordinates between stations should match the allocation of row and column indexes inside the matrix. In other words, created matrices slightly mimic the spatial representation of unbalanced stations in the real world.

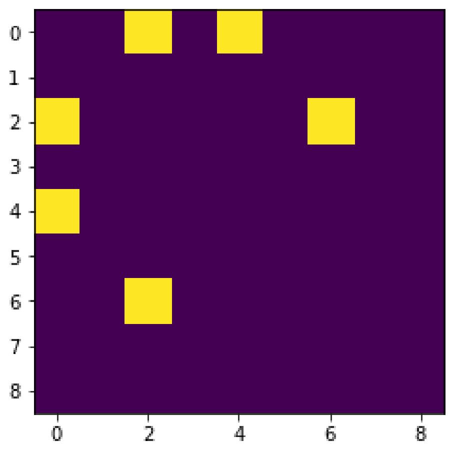


Figure 24: Adjacency matrix for March 2018 with isolated stations for that year

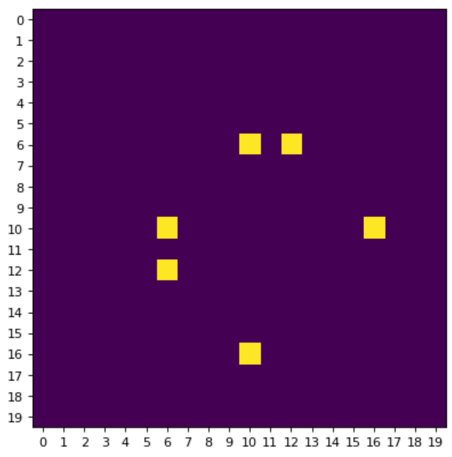


Figure 25: Adjacency matrix for March 2018 combined with stations from both years

For example, in Image 24 there is a matrix representing a particular month (March of 2018 in this case) with the most unbalanced flows being between stations (0,2), (0,4) and (2,6). Stations 0,1,2,3,4 and 5 are located in the center of Cambridge, station 6 is located in Boston Bay area, 7 and 8 are found in central Boston. This means that, depending on the pattern of pixels in the matrix, there are different flows connecting distinct neighbourhoods. Since CNN is a model that learns to recognize these patterns and tries to find them in pixelated images that have not been observed before, that is the main reason why the method had been chosen for this purpose.

When we want to observe a two year period of unbalanced snapshots instead of one year period, of course that means that we will probably have some new unbalanced stations appearing in the unbalanced pairs. Thus, our matrices will grow in order to accommodate this new stations. In Image 25 we can see the same stations as in Image 24 but with an expanded matrix which means that encoding of the stations with their identification numbers will be slightly different, but the spatial configuration pattern will never lose its relative shape. The specific way that 20x20 matrix has its space configured and defined based on the location of station spatially across the area can be seen in Figure 26. Three cities are found sharing the matrix: Somerville, Cambridge, and Boston. Cambridge has certain concentration of observed unbalanced stations around Inman Square area, Central Square area, shore area of Cambridgeport, and Campus area of MIT. Boston has a Port Bay area just across the Harvard bridge, North Boston consisting of West End and central Boston area, South Boston between Waterfront and Financial district.

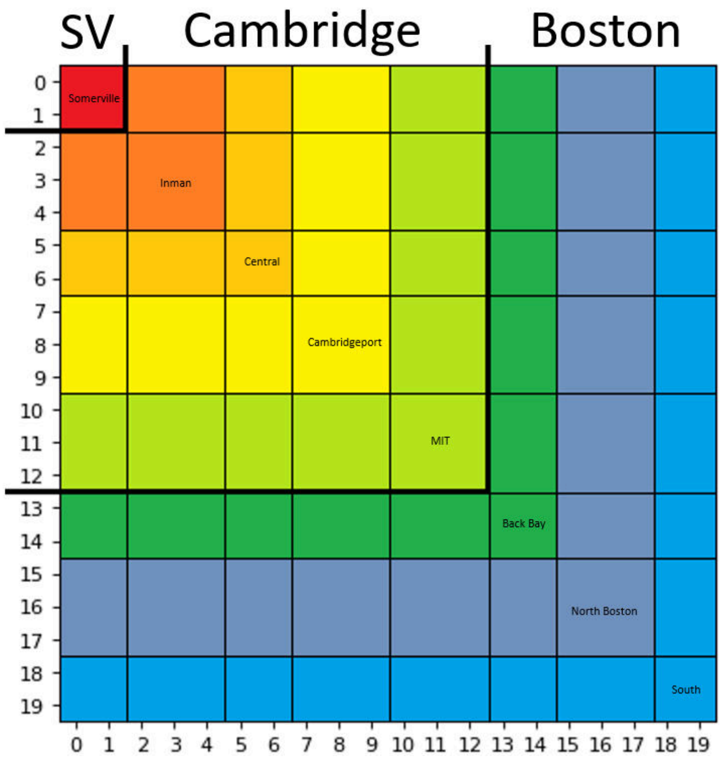
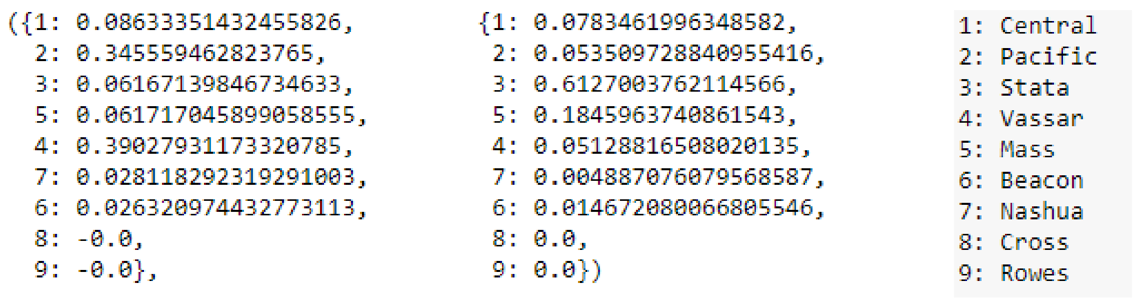


Figure 26: 20x20 matrix configuration depending on spatial position of stations

6.2 Graph Embeddings

In this subsection, a short description of some additional graph analysis we are able to perform on a bike usage dataset will be described. An example of a an aggregated visualization can be seen in Figure 28.

Firstly, we can perform Hyperlink-Induced Topic Search (HITS) algorithm which is a link analysis algorithm that rates hubs and authorities of a given network. For instance, if we take year 2018 as an example, in Figure 27 we can see the obtained hub scores on the left and authority scores in the middle. In this particular case, node number 4 (Vassar bike station) is the best hub, meaning the node that has the most bike check-ins coming from a variety of different stations. Likewise, node number 3 (Stata bike station) is the best authority, or the node that has the most bike check-outs directed towards many other stations.



```
({1: 0.08633351432455826,
  2: 0.345559462823765,
  3: 0.06167139846734633,
  5: 0.06171704589905855,
  4: 0.39027931173320785,
  7: 0.02811829231929100З,
  6: 0.02632097443277311З,
  8: -0.0,
  9: -0.0},
```

```
{1: 0.0783461996348582,
 2: 0.05350972884095541б,
 3: 0.6127003762114566,
 5: 0.1845963740861543,
 4: 0.05128816508020135,
 7: 0.004887076079568587,
 6: 0.014672080066805546,
 8: 0.0,
 9: 0.0})
```

```
1: Central
2: Pacific
3: Stata
4: Vassar
5: Mass
6: Beacon
7: Nashua
8: Cross
9: Rowes
```

Figure 27: Hub (left) and Authority (middle) scores for 2018

Additionally, a method called spectral clustering can be utilized. In general, clustering is an unsupervised learning method and algorithm that operates uses a similarity measure to cluster data points together. In spectral clustering, the data points are treated as nodes of a graph, and clustering is treated as a graph partitioning problem. Spectral clustering consists of three steps: creating a similarity graph, computing the first k eigenvectors of its Laplacian matrix to define a feature vector for each object, and running a k-means algorithm on these features to separate objects into k classes.

In our implementation, we will consider year 2018 as an example. For spectral clustering method, we will use size of cluster 2, precomputed affinity, and a number of times the k-means algorithm will be run with different centroid seeds set to 100. The result show that first five stations (Central, Pacific, Stata, Vassar, Mass) are part of one cluster and last four stations (Beacon, Nashua, Cross, Rowes) are part of another cluster.

In a second implementation, we can change to assign labels using an approach called discretization, which is less sensitive to random initialization. This is used instead of k-means algorithm. The results with this approach show that All the stations are part of the first cluster, expect for Cross and Rowes which are part of the second cluster. A visualization of this specific case can be observed in Figure 28.

Another possibility could be to increase the number of clusters by one. Using discretization method, we are able to get the following results:

cluster 1: Central, Vassar, Nashua
cluster 2: Pacific, Stata, Mass, Beacon
cluster 3: Rowes, Cross

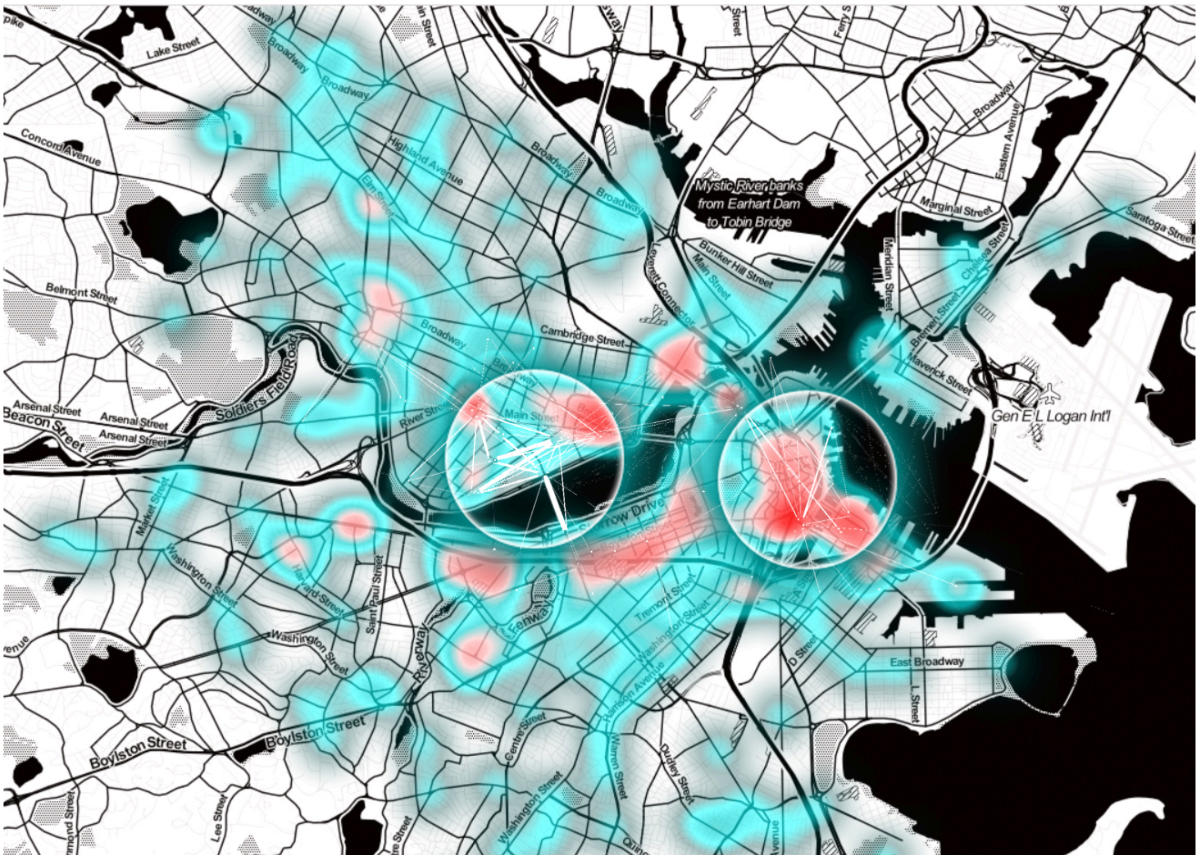


Figure 28: Aggregated heat map with directed graph of mobility flows showing most used areas for July 2018

Using spectral clustering approach could be possibly used as a means to assign identification umber to different bike stations, apart for just being a visualization method.

Also, before running the CNN, we can explore prepared image data using t-SNE and PCA algorithms.

As described in theoretical background chapter, T-Distributed Stochastic Neighbor Embedding is a non-linear technique for dimensionality reduction. It attempts to find patterns in the data by identifying observed clusters based on similarity of data points with multiple features. This method is mainly used a data exploration and visualization technique.

Principal Component Analysis or PCA is a linear feature extraction technique. It combines input features in a specific way that it is possible drop the least important feature while still retaining the most valuable parts of all of the features.

An example of such visualization techniques for CNN training data for the year 2018 can be observed in Figure 29. Input data is training set consisting of each month in the year 2018. There are less labels than there are months because some of the months have the same label. Also, training data is repeated five times rendering the size of the entire dataset to be 60 instead of just 12.

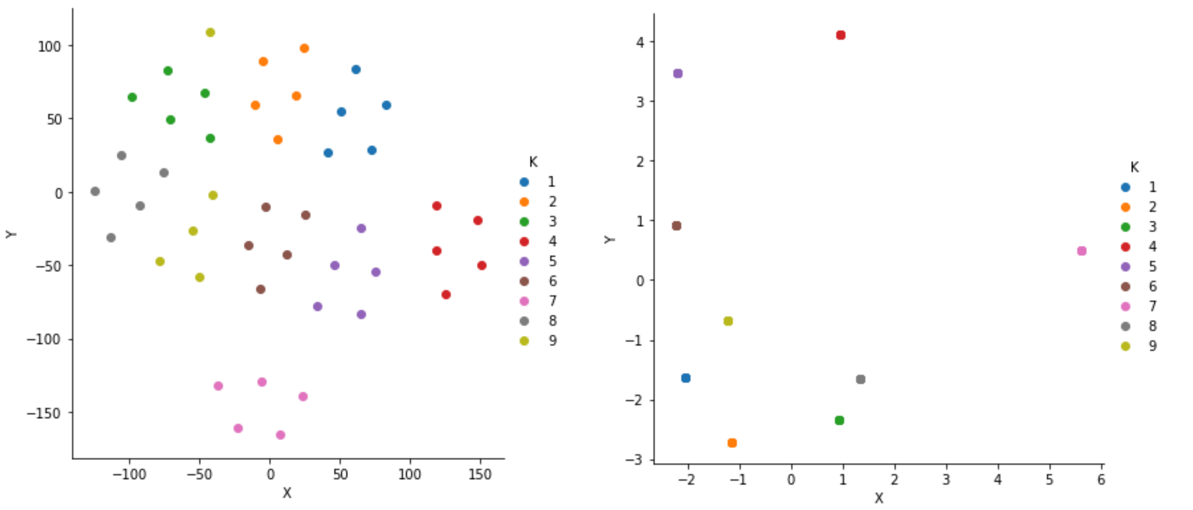


Figure 29: clusters of t-SNE (left) and PCA (right) for 2018 data

6.3 Motivation

Here, a rationalization for using this particular method with CNN will be explained. Primarily, unbalanced bike flows between stations are a real problem, which is especially becoming more prominent as BSS grows bigger within the certain area over time [59]. However, it is not easy to predict exactly which unbalanced pairs are to be expected to emerge, but they can be approximated using the current state of BSS, which will be used as a test data for CNN. There are also some trends that can be noticed, like having one smaller graph in the winter and two subgraphs in the summer: a giant component in Cambridge and a smaller component either in Boston or Somerville. Weighted diameter of the giant component is smaller compared to the components outside Cambridge.

Moreover, not only are the unbalanced pairs of nodes also a good approximation of the most used bike station in general (as described in the Chapter 4) but looking at unbalanced pairs can detect some less used bike stations that suddenly during a short period of time get congested with bike traffic and build their unbalanced ratio with some other station to a critical point. Also, CNN is needed as a step before RNN simply because it is not feasible to try and predict the most unbalanced pairs with RNN method alone. It is not because it wouldn't be possible but because graphs, although seemingly simple, can require a vast computing power in order to predict bike flow dynamics for every single pair of nodes [60]. Having a number "n" of nodes, the total number of possible edges is "E" as observed in the edge formula below. Assuming 10 seconds which, on average and with the most simplest method, RNN takes to predict time series, it would take 10*e seconds. Number of stations being 194 in 2018 means that it would take total of 187'210 seconds (52 hours). After that, we would still need to calculate flow differences and order them in descending order. As we would like to make a short-term prediction for the upcoming couple of days or weeks, any method where we would use exclusively RNN is absolutely not appropriate from the computational complexity point of view.

$$E = \frac{n * (n-1)}{2} \tag{6.1}$$

6.4 Convolutional Neural Network

Prepared training data, which is being provided to CNN as an input (see Figure 30), consists of 24 pixelated images with size 20x20. Each image represents the top 3 most unbalanced pairs of stations for one particular month. Yellow pixels, as seen in Figure 25, is an indication of an edge between the station "i" and "j", where "i" and "j" are identification numbers of those two stations. Not only is it an indication of an edge but an unbalanced link between them. Training data will be used for CNN to observe and learn from all past configurations and try to guess which of the pre-existing shapes would match the best each of the test data sets we will additionally provide. It is important to notice that a small number of months in the training set have duplicates because some months have had the same top 3 unbalanced station pairs. This duplicates are a valuable indication for CNN that this particular configuration may have a greater importance and is more likely to be repeat again in the future. Each one of the training images is manually labelled with a certain label number which corresponds to its unique spatial configuration. Of course, duplicates will be assigned an identical label as their spatial configuration is the same.

The structure of CNN used for this project can be observed in Table 13. This specific configuration was modified and inspired by CNN used for performing on National Institute of Standards and Technology database (MNIST) dataset, which uses handwritten digits for training various image processing systems.

Table 13: Configuration of Deep CNN

Type	**Structure**
Input	24 x 20 x 20
Conv	filter 12 x 5 x 5, stride 2 x 2, ReLu
Pool	Max, 2 x 2
Dropout	p = 0.12
Conv	filter 24 x 5 x 5, stride 2 x 2, ReLu
Pool	Max, 2 x 2
Conv	filter 48 x 5 x 5, stride 2 x 2, ReLu
Pool	Max, 2 x 2
FC	120, ReLu
Dropout	p = 0.5
FC	84, ReLu
softmax	21

Convolutional layers consist of a specified amount of convolutional filters applied to the image. These layers perform mathematical operations and an output feature map is given as a result. Typically, ReLU activation function is used for the output as it converges faster.

Pooling layers reduce the dimensionality of the feature map which decreases processing time. A commonly used pooling algorithm is max pooling, which extracts 2x2-pixel tiles, keeps their maximum value, and discards all other values.

Dense or fully connected layers perform classification on the features after convolutional and pooling layers have been used. In a dense layer, every node in the layer is connected to every node in the previous layer.

Dropout rate refers to ignoring neurons during the training phase. During that phase, a certain set of neurons, chosen at random, is either dropped out of the network with probability 1-p or kept with probability p. Because a fully connected layer occupies most of the parameters, neurons develop co-dependency between each other during training process, which leads to over-fitting of training data. In short, dropout helps us prevent the over-fitting. This is especially useful for the project described in this thesis as we are using a repeating training set and dropout rate will be used to balance the over-fitting created by such an approach.

Softmax is implemented through a neural network layer just before the output layer. The Softmax layer must have the same number of nodes as the output layer. This means that the number of layers must correspond to the number of labels we have defined. In this case, that is exactly 21 layers or 21 different layers. Also, to clarify why instead of 24, which is the number of non-repeating training images, we have 21 label. This is because some of the images in our training set are repeating multiple times and hence, they are assigned the same label. In conclusion, that is why the number of labels is smaller than the number of images in the non-repeating training set, and is the reason behind the number of Softmax layers.

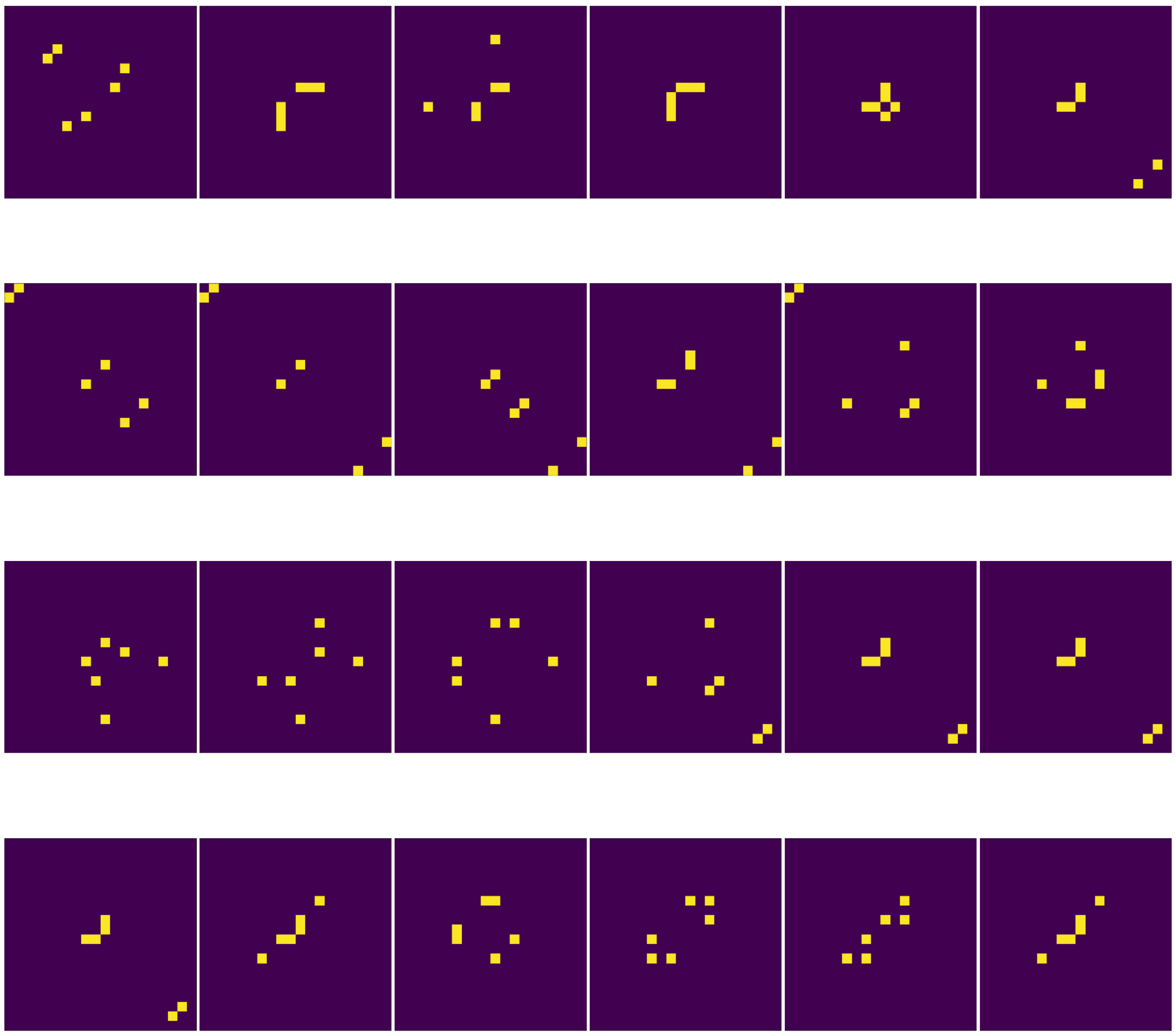


Figure 30: Training data

Regarding the test set, as we want to evaluate how CNN performs, a combination of various spatial configurations will be used. Just as a base case, every label from the training set will be used in the test set as well. This is because we want those labels to be predicted with a 100% accuracy as they are identical to one of the shapes that we already know. In addition, new shapes which have never been observed before will be put into the test dataset to see how CNN can handle those. As seen in Figure 31, first four rows consist of permuted training images which are already known. Last two rows have a combination of images with only two pairs of stations that resemble an existing pattern, two pairs of stations that are found in a new, never before seen relationship, and image with more than 3 pairs of stations that both have some older edges, but also new ones.

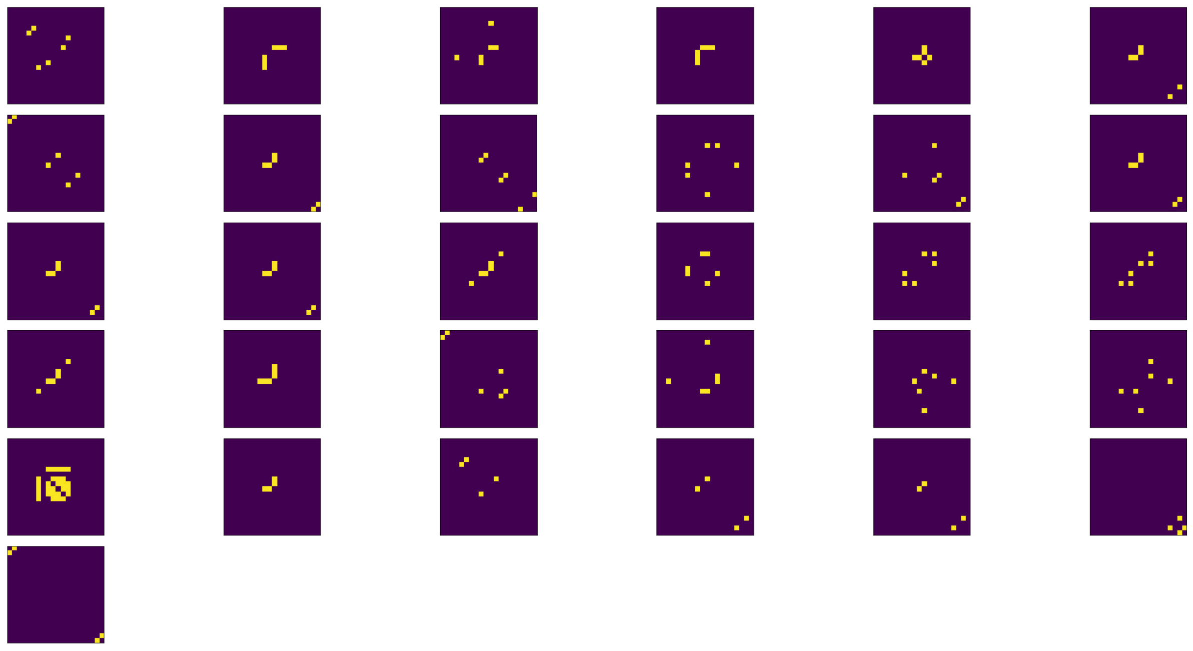


Figure 31: Test data

Before running CNN, we should define a ground truth labels for our new shapes inside the test set. That is required in order to evaluate the precision of CNN label prediction. Some of the new shapes resemble not one, but multiple training images. This is why, once we evaluate how CNN performed, any false prediction will be inspected closely to determine whether there are somewhat precise even though our ground truth label was not matched. Also, our training dataset for monthly granularity is not sufficiently big enough for producing high enough accuracy, but that can be fixed by expanding it simply by duplicating existing 24 training images 5 times resulting in 120 images total. It is not advised to duplicate more than that as there could be a danger of overfitting [61]. CNN used for this purpose had 3 convolutional layers, 2 drop-out layers and 2 fully connected layers. Each convolutional layers had 12, 24 and 48 hidden layers within. Optimizer used was adam with the learning rate 0.001, activation function utilized was rectified linear unit, and the number of epoch was set to 100. Again, the configuration can be closely observed in Table 13.

As a final result, the precision is 93.5483 % by predicting a correct label for 29 out of 31 images. But, counting only images with completely novel shapes, precision achieved is 0.714 % or, 5 out of 7. It is important to highlight that falsely predicted labels occurred for those shapes which had completely new edges and some extreme edge combinations.

Also, they were rendered false only because they did not match completely to the ground truth, and the ground truth was manually chosen, hence being also prone to error. In a sense, falsely predicted labels were still good, especially because spatial restriction was adhered to, meaning that CNN never picked a label that had some random edges in a far away place but tried to match the dispersion of pixels as close as possible. This is exactly possible because of the adjacency matrix design and why it was crucial for the prediction part. For example, if the novel shape has a matrix representing all the possible connections in Cambridge exclusively, the predicted matrix (although not matching the ground truth) will have its three connections only in Cambridge area as well because CNN learns that other faraway connections (Somerville or Boston) are more rarely observed in combination with Cambridge links.

In case we only used only 24 training images instead of 120, this would render out precision to 61.29 % (19 out of 31). And counting only novel shapes, all of them would be false.

In Figure 32, accuracy through epochs can be observed. Yellow line represents validation accuracy reaching 100% and red line represents test accuracy reaching, already before mentioned, 93.5483 %. On the other hand, in Figure 33, loss function is depicted. Magenta function shows how validation loss progresses, and green function shows test loss.

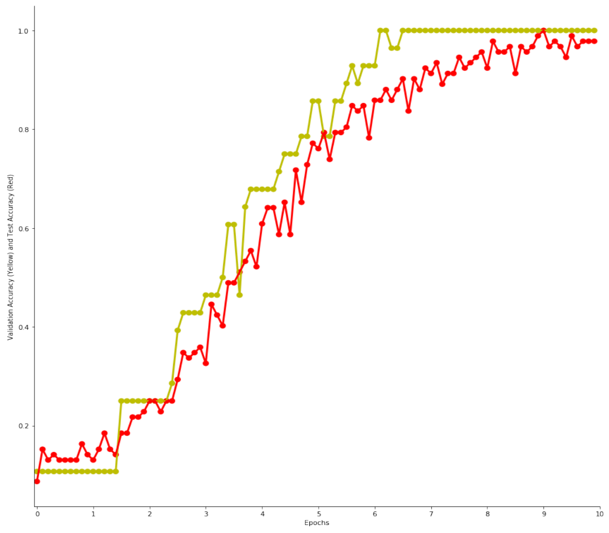


Figure 32: Validation (yellow) and test (red) accuracy

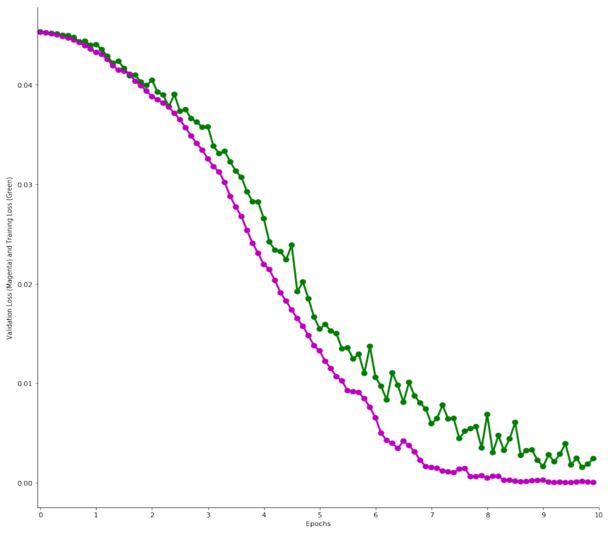


Figure 33: Validation (magenta) and test (green) loss

An addition to improving CNN in a segment of deciding which station configuration or specific station links might have greater importance before deciding to label the test image is to add weights to each pixel in the image representing bike usage for that month. In that case, our training set of matrices gets each pixel coloured based on the intensity of asymmetrical balance between certain stations. For example, higher asymmetrical weight would correspond to a higher intensity color and thus, be considered as more important than maybe similar image configuration with lower weight values. This method was not fully utilized here because in order for it to show any benefits, we would need a much larger training set consisting of many years worth of data. However, this could be an interesting approach if such a method would be attempted on a much larger BSS, or scaled down to a neighbourhood level. In Figure 34, such weighted approach can be seen depicted for the previously used training set. Lighter yellow and green colours represent higher unbalanced difference, while darker green and blue colours mean that the unbalanced difference has an increasing lower value.

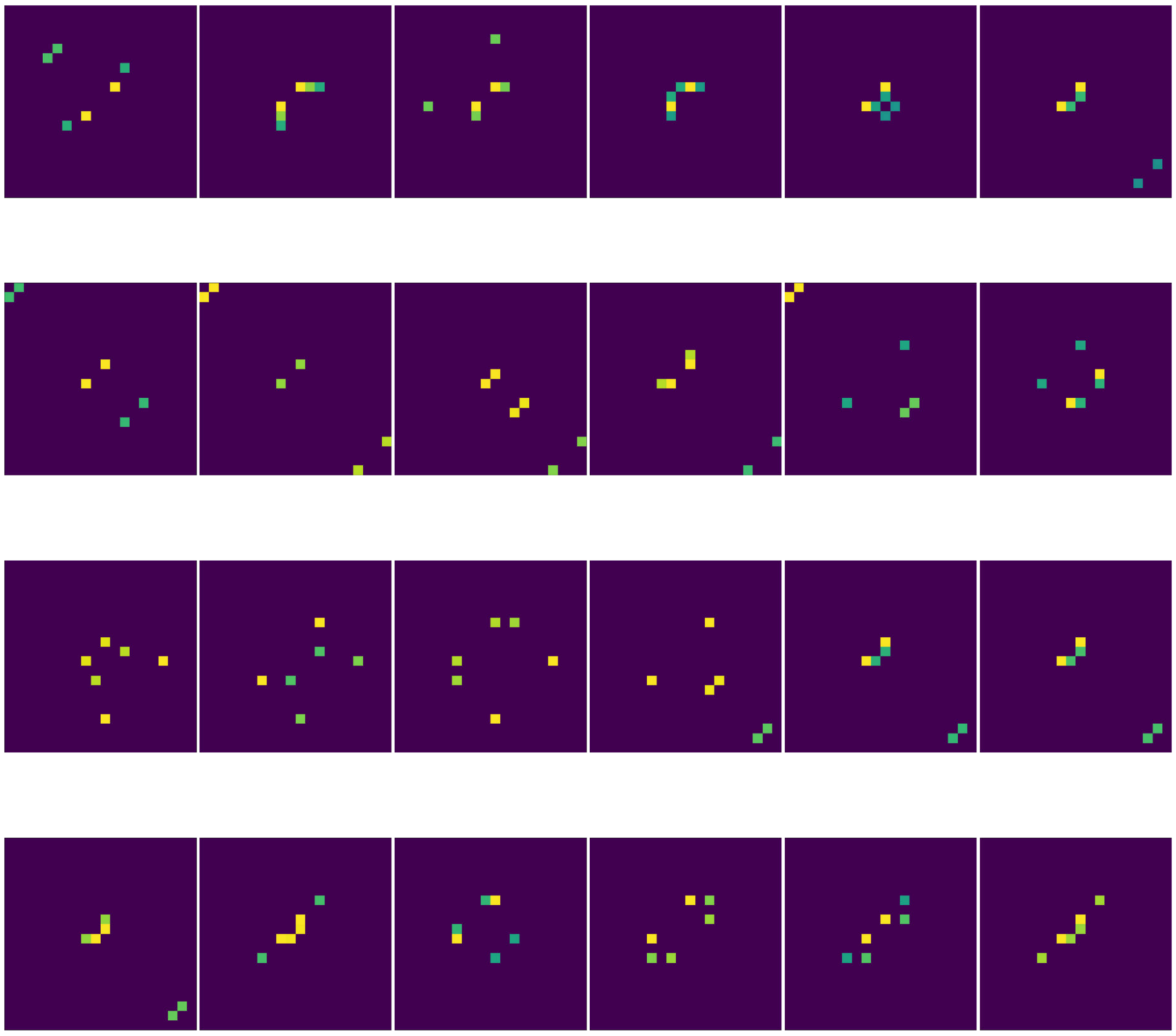


Figure 34: Weighted training set

6.5 Adding Candidate Novelty Links

In order to strengthen the achieved CNN prediction, we will carefully consider adding a possible surprise link to the predicted configuration, which has a highest probability of emerging as a future most unbalanced link in addition to the ones already contained within the predicted label. Only those links that have been fairly recently emerging close, but not quite to the top 3 most unbalanced spots, can be considered as candidates for this analysis. Additionally, one of the top 3 unbalanced candidates can be considered as well in case we observe that CNN completely skipped it during label prediction. This happens because of the small training dataset but using this method mitigates such problem.

A model chosen for predicting such a candidate is called convolutional multivariate LSTM model. Although this model is capable of vector output, we will only consider the first predicted item because multi-step prediction becomes very inaccurate further into the future the model tries to extrapolate. For example, if we have seen that a specific unbalanced station pair had been showing up as the 4^{th} most unbalanced link, there could be a probability that such a link would emerge into the top 3 list in some close future moment. Therefore, we will inspect all of the past monthly unbalanced differences this link ever achieved and extract it as an array of integers. For instance, this could be a tuple consisting of station 10 and station 11, and in that case its array for year 2017 and 2018 is following:

[13, 5, 32, 75, 108, 118, 53, 87, 140, 77, 45, 15, 32, 31, 27, 11, 96, 115, 172, 195, 31, 33, 4, 106]

Now, we can utilize multivariate convolutional LSTM model to predict what could be the next possible array item, based on all the previous ones observed.

The model predicts that the next item should be 137, while the ground truth for January 2019 is 108. That renders an acuracy of around 78.8%. When run again, predictions were pretty consistent: 122.62, 114.55, 119.35, 119.64. On average we are getting prediction of value 122.64 which leaves us with a satisfactory accuracy score of 88.05%. Although this model does not guarantee high accuracy, it is still a good indicator if some of the less unbalanced links are having a growing trend to become the next top 3 candidate and exact value is not needed.

The motivation behind choosing this particular method is because it has a low runtime of just 2.64 seconds on average, has a rather high accuracy, and is able to predict an arbitrary number of future step, although for the purpose of this thesis only one additional future step will be predicted in order to keep accuracy score as high as possible. This is because having monthly granularity cannot handle extreme changes in between the steps and still give a satisfying output.

Although CNN LSTM method has a slightly better result, its average runtime is almost doubled (5.9 seconds) compared to convolutional LSTM method. It is, then, justified to consider using convolutional LSTM as the aim of the methodology is to be as computationally lightweight as possible, and the difference in runtime might be significant in case we have to handle larger arrays.

Convolutional LSTM has a unique architecture utilized for handling arrays through 1-D convolutional layer, pooling layer, dense layer,and LSTM layer. In particular for this case, 64 convolutional filters are used, 50 LSTM hidden nodes, ReLU activation functions, and 500 epochs.

In Table 14, a short overview of all the forecasting methods considered for the novelty link discovery can be found. It is quite clear that only a combination of an architecture that consists of convolution layers and LSTM layers is relevant enough.

Table 14: Evaluation of novelty link methods

Method	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Average**	%
Stacked LSTM	125.27	168.86	137.92	176.39	92.98	140.28	77
Conv LSTM	137.06	122.62	114.55	119.35	119.64	122.64	88
Vanilla LSTM	129.18	135.40	117.44	120.79	125.24	125.61	86
Bidirectional LSTM	123.12	130.67	141.87	179.11	145.43	144.04	75
CNN LSTM	121.40	121.43	139.16	113.51	109.82	121.06	89

To summarize this section, LSTM can be used to model univariate or multivariate (with or without multi-step) time series forecasting problems. A Vanilla LSTM is an LSTM model that has a single hidden layer of LSTM units, and an output layer used to make a prediction. A Stacked LSTM has multiple hidden LSTM layers stacked one on top of another. A Bidirectional LSTM can learn the input sequence both forward and backwards and concatenate both interpretations. It is implemented by by wrapping the first hidden layer in a wrapper (bidirectional) layer. A CNN LSTM is a hybrid CNN model with an LSTM back-end where the CNN is used to interpret subsequences of input that together are provided as a sequence to an LSTM model to interpret. A Conv LSTM is a model where convolutional reading of input is built directly into each LSTM unit. Difference between CNN LSTM and Conv LSTM is that Conv LSTM is an LSTM whose gates perform convolutions, and CNN LSTM refers to an architecture where LSTM is stacked on top of CNN.

7 Discussion

7.1 Results

In this section, all of the previously introduced prediction methods from Chapters 4 through 6 will be applied for the Boston use case in order to predict spatial structures and dynamic patterns. For this purpose, months of January, February, March and April of the year 2019 will be used as a validation test set to discuss the overall performance of this thesis project.

7.1.1 Prediction Results for January 2019

First, January (69'872 trips in total) will be taken and split into two sets. Smaller (training) set will consist of the time period when 1/3 of bike trips were cycled, generally approximated by the first ten days of January and for those first 10 days, a top 3 most unbalanced pairs of stations will be produced and transformed into adjacency matrices (see Figure 35). In case we encounter a new station never observed before within the list of top 3 most unbalanced pairs obtained, that pair is ignored (or a known station in the pair is represented as a loop pixel in the adjacency matrix). Of course, in an ideal case, new stations should be added and matrices expanded as those stations could start appearing in a top 3 list for future months and years. This is also true in case top 3 lists would be expanded to top n lists, where n represents the number of tuples examined. Such thing is feasible but would require manual expansion and filling in matrices. Here, an analysis for only top 3 most unbalanced pairs of stations will be presented.

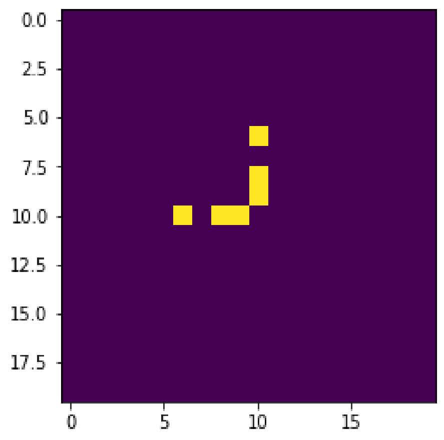


Figure 35: First 10 days of January 2019 unbalanced links: (10,11), (11,7), (11,9)

CNN will try and classify the matrix representing the first week of January to the one of the existing labels from the previous 2 years contained in our training set of matrices. This predicted label will be used as a mobility flow model for the whole month of January as we assume that identified unbalanced flows for the first third of the month usually tend to converge to the predicted labels for the rest of a month. Predicted configuration can be observed in Figure 36.

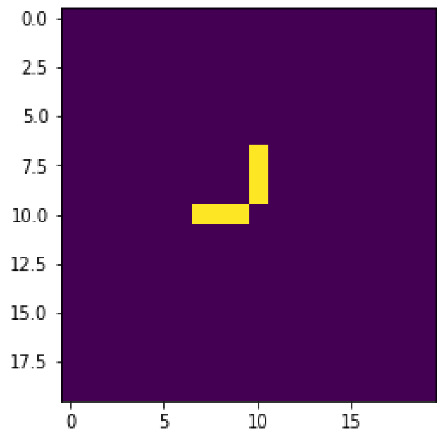
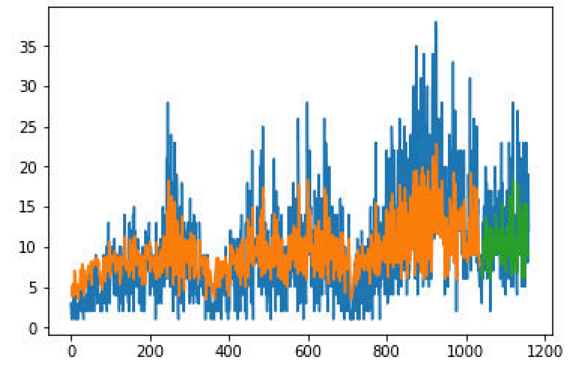


Figure 36: CNN prediction for the whole month of January 2019: (10,11), (11,8), (11,9)

Next, for station pairs that are found in the predicted matrix, RNN method is utilized as we want to find the predicted flow dynamic for each one of the station pair. To get the most unbalanced scores, we subtract the predicted flows from the unbalanced pairs, one tuple at a time. When using RNN to predict flows between station Pacific (10) and station Stata (11), we considered bike flow data ranged from January 2016 to April 2019, and accordingly, the training data consisted of 0.8945% of the total data to match that the test data fits with the whole year of 2019, which is 120 days for first 4 months. RNN settings used included: 400 epochs, 12 ReLU activation functions in the first hidden layer and 8 tanh activation functions in the second hidden layer. Prediction results achieved for the last 21 days (three weeks) of January were 78.38%. Cumulative sum of the bike usage during those two weeks was 241.05843 predicted, and the ground truth is 251. This is for the bike traveling from Pacific station (10) to Stata station (11).

Now, the same thing needs to be done in the reverse flow direction, from station Stata (11) to station Pacific (10). For this flow, we are getting a prediction with 77.43% accuracy that the cumulative sum of bike usage between January 10 and January 31 will be 156.40091, while the ground truth is 162. The RNN prediction pair output can be observed in Figure 42.



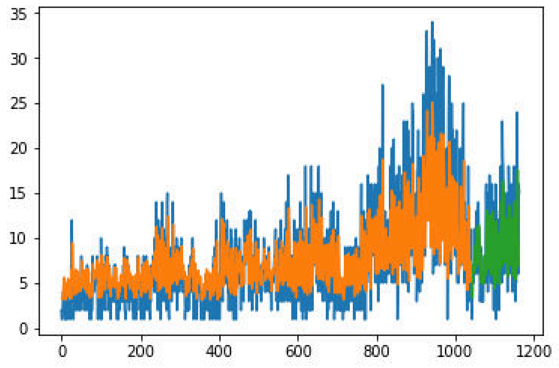


Figure 37: Pacific to Stata flow (left) and Stata to Pacific flow (right)

Finally, we can now observe what we predicted and see that the difference of flows between station 10 and 11, being 156.40091 subtracted from 241.05843, is 84.65752 bikes (rounded to 85), thus making the station Pacific more sparse with bikes. The difference for the ground truth was calculated by observing the real captured data from January 10 to January 31. The observed value for this difference is 68, and the precision score for predicting this asymmetry is 80%.

Same procedure is repeated for other station pairs.
Going from station Vassar (9) to Stata (11) station the predicted value is 219.41492, while 186 is an actual value. This renders the accuracy to be 84.77%. From station 11 to station 9, we get predicted value of 145.681, while 127 is as an actual value. Hence, accuracy is 87.17%. The predicted difference between the flows is 73.73392 (rounded to 74) and compared to the real value of 61, the overall accuracy is 82.43%.

Regarding the station pair Stata (11) and Sidney (8) that CNN gave as a result instead of Stata (11) and Central (7), the difference is unnecessary to be predicted or calculated as it is almost zero, hence we got a false result for this link prediction. In this specific case, getting a zero value is informing as that CNN predicted a link that is not active during this time of the year.

Still, we can possibly add a novelty link or retrieve a link overlooked by CNN. That can be done by observing which unbalanced pair is appearing in the top 4 most unbalanced links and in this case we can clearly see that the CNN overlooked an unbalanced flow between Stata (11) and Central (7).

Going from station 11 to station 7 the predicted value is 134.01521, while 136 is an actual value. This renders the accuracy to be 98.54%. From station 7 to station 11, we get predicted value of 82.25274, while 69 is as an actual value. Hence, accuracy is 83.88%. The predicted difference between the flows is 51.76247 and compared to the real value of 67, the overall accuracy is 77.25%.

7.1.2 Prediction Results for February 2019

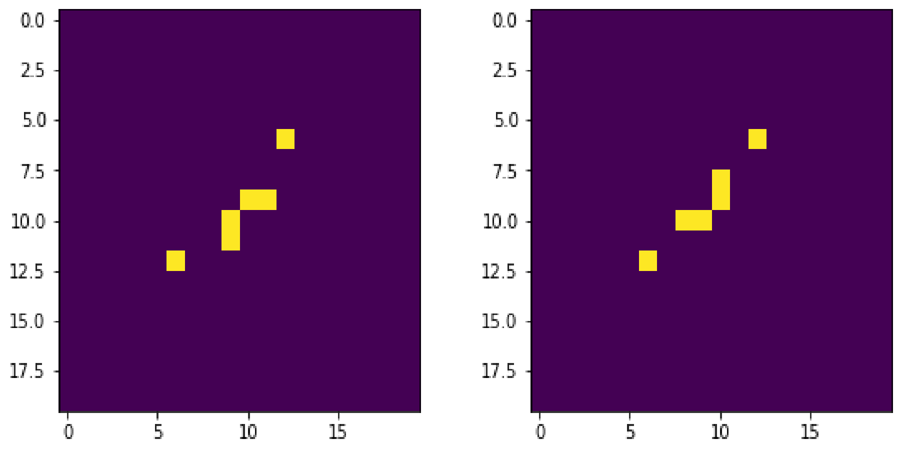
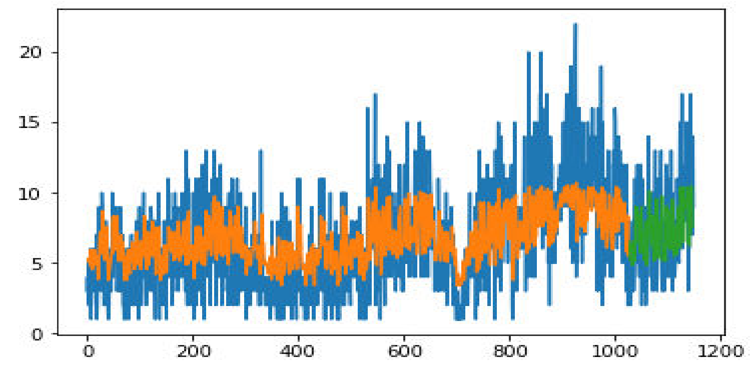


Figure 38: First 1/3 flows of February (left) and predicted rest of February (right)

Predicted unbalanced pairs of stations for February are (11,9),(10,11) and (13,7). These pairs can be observed in Figure 38. Using RNN on these pairs we get the following flow differences:

Predicted flow of a tuple (13,7) is 148.43597 and its real flow is 146, amounting to an accuracy of 98.35%.
Predicted flow of a tuple (7,13) is 90.60720 and its real flow is 76, meaning that the obtained accuracy is 83.87%.
These flows can be observed in Figure 39.
Predicted flow difference is 57.82868, while the ground truth is 78. The overall difference accuracy is 74%.



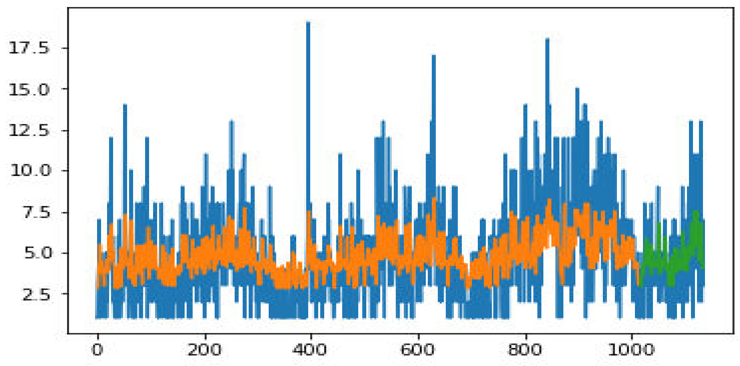


Figure 39: (13,7) Mass to Central (top) and (7,13) vice-versa (down)

Directed link (11,9) has a predicted flow 163.80017 and real flow 183, accounting for an accuracy of 89.5%.
Directed link (9,11) has a predicted flow 226.40657 and real flow is 222, amounting to an 98% accuracy.
Predicted difference between the flow of two directed links is 62.6064, while the ground truth is 39. The overall difference accuracy is 62%. The low accuracy score we got is the consequence of falsely predicting this station pair when using CNN.

Directed link (10,11) has a predicted flow 223.32707 and real flow is 268. Obtained accuracy is 83%.
Directed link (11,10) has a predicted flow 186.0166 and real flow is 216. Obtained accuracy is 86%.
Predicted difference between the flow of two directed links is 37.31047, while the ground truth is 49. The overall difference accuracy is 76%. However, it is important to mention that this CNN predicted pair is not found in the top 3 most unbalanced ground truth stations for the rest of February.

Also, using a novelty link method does not help us to uncover the never before observed station named “Deerfield”. This is because the station starts creating the unbalanced link with “Stata” only in the 2/3 of the month and that is we are unable to detect it early.

7.1.3 Prediction Results for March 2019

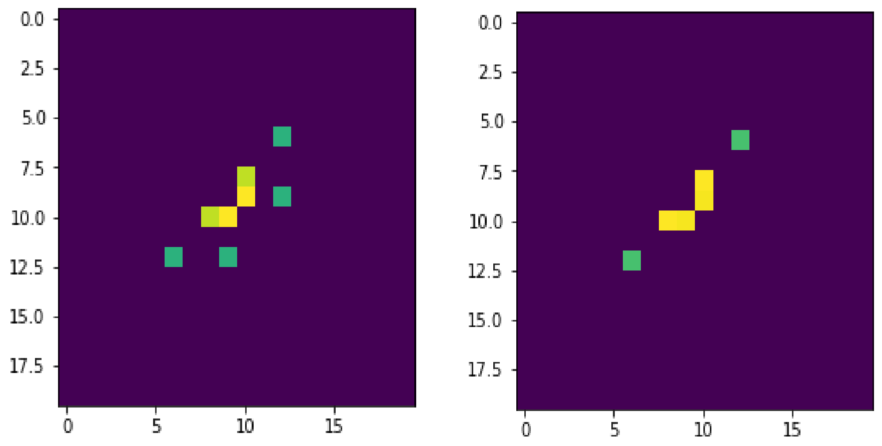


Figure 40: First 1/3 flows of March (left) and predicted rest of March (right)

Predicted unbalanced station pairs for March include tuples: (11,9),(10,11) and (13,7). Visualization of tuples can be observed in Figure 40. Just to demonstrate, this CNN prediction visualization technique used weighted colouring, but it does not make any difference in this thesis due to a small number of training sets. We will be using RNN prediction method on these particular pairs retrieved, in order to get flow differences.

Directed link (13,7) has a predicted flow 124.50949 and real flow 125, accounting for an accuracy of 99.6%.
Directed link (7,13) has a predicted flow 78.81064 and real flow 73, amounting to an accuracy of 92.6%.
Predicted difference between the flow of two directed links is 45.69885, while the ground truth is 64. The overall flow difference accuracy is 71.4%. This most unbalanced pair flow calculated above exists as a ground truth in the top 3 most unbalanced stations for the rest of the March.

Ground truth top 3 most unbalanced tuples for the rest of the March are (13,7),(7,11), and (17,11).

Unfortunately, using a multivariate RNN method for adding novelty links does not seem to work for March, either. This might be because March and February are considered as transitional months in New England area between winter and spring. That also indicated that many new links will appear in the second part of the month which our methods are not able to predict.

7.1.4 Prediction Results for April 2019

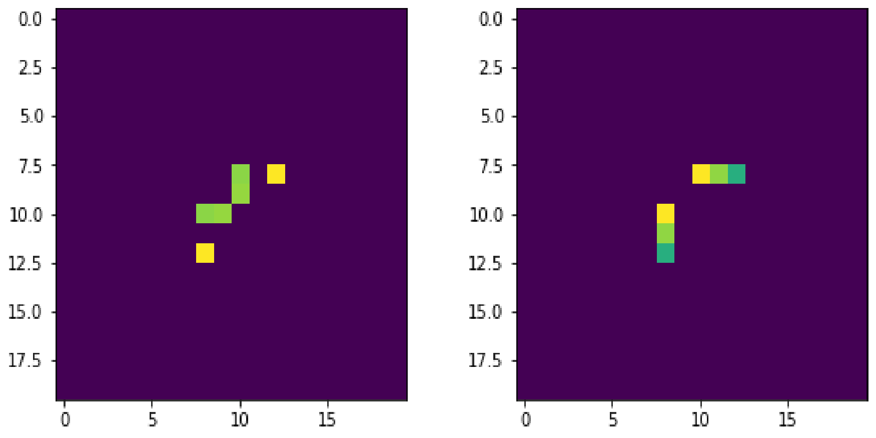


Figure 41: First 1/3 flows of April (left) and predicted rest of April (right)

Predicted unbalanced station pairs for April include tuples: (9,11),(9,10) and (9,13). Visualization of tuples can be observed in Figure 41. Here, as well, just for demonstration purposes, CNN prediction visualization technique with weighted colouring was used. We will be using RNN prediction method on these particular pairs retrieved, in order to get flow differences.

Directed link (11,9) has a predicted flow 131.11469 and real flow 137, accounting for an accuracy of 95.7%.
Directed link (9,11) has a predicted flow 239.93716 and real flow 261, amounting for an accuracy of 91.9%.
Predicted difference between the flow of two directed links is 108.82247, while the ground truth is 124. The overall difference accuracy is 87%. This unbalanced pair calculated above exists as a ground truth in the top 3 most unbalanced stations for the rest of April.

Directed link (9,13) has a predicted flow 179.53601 and a real flow 194, accounting for an accuracy of 92.5%.
Directed link (13,9) has a predicted flow 233.67902 and a real flow 260, amounting for an accuracy of 89.8%.
Predicted difference between the flow of two directed links is 54.14301, while the ground truth is 66. The overall difference accuracy is 82%. This unbalanced pair calculated above exists as a ground truth in the top 3 most unbalanced stations for the rest of April.

Directed links (9,10) and (10,9) are not in the top 3 ground truth most unbalanced stations for the rest of April, even though our CNN model suggests that it could be. if we try to utilize the same RNN prediction method, we will retrieve an extremely small bike flow between Vassar and Pacific, close to zero value, which makes it quite obvious that this particular tuple is not our unbalanced candidate.

Ground truth for the rest of the April is (9,11), (7,13), and (9,13) as top 3 most unbalanced tuples.

Moreover, we can utilize a multivariate RNN method for adding a novelty link for the predicted configuration. If we observe the 4^{th} most unbalanced link for the first 1/3 of all monthly flows, we can successfully notice that the tuple in question is (7,13). The method is, naturally, hinting towards a stable, relatively high, trend of that specific pair. This means that we can add the link to our flow prediction analysis. Specifically, observed array of unbalanced flows for tuple (7,13) in 2017 and 2018 is following:

[41, 28, 1, 39, 28, 23, 28, 50, 70, 55, 68, 36, 30, 77, 69, 96, 59, 64, 68, 140, 88, 123, 103, 108].

Next predicted array item with multivariate RNN is 70.61995. Although the trend is not growing, the predicted value is still higher than a directed link (10,9) acquired through CNN method, whose predicted unbalanced value difference was 54. Therefore, we can confidently take (7,13) and add it to our analysis.

Directed link (7,13) has a predicted flow 91.315445 and a real flow 115, accounting for an accuracy of 79.4%.
Directed link (13,7) has a predicted flow 157.36572 and a real flow 183, amounting for an accuracy of 86%.
Predicted difference between the flow of two directed links is 66.050275, while the ground truth is 68. The overall difference accuracy is 97%.

7.1.5 Overall prediction results

In the Table 15, evaluated results for all four moths in 2019 can be viewed. Average RNN accuracy is calculated as a mean value between RNN top link scores. Accuracy scores are expressed in percentages, where CNN is calculated as number of true predictions out of total 3, and RNN is calculated as a quotient between predicted unbalanced difference and observed unbalanced difference.

Table 15: Evaluation of results

Method	**Jan 2019**	**Feb 2019**	**Mar 2019**	**Apr 2019**	**May 2019**
CNN Top 3 Score	66.7%	33.3%	33.3%	66.7%	0%
Multivariate CNN Score	100%	33.3%	33.3%	100%	66.6%
RNN Top 1 Link Score	77.2%	74%	71.4%	87%	64%
RNN Top 2 Link Score	80%	n/a	n/a	82%	n/a
RNN Top 3 Link Score	82.4%	n/a	n/a	97%	87%
Average RNN Accuracy	78.8%	74%	71.4%	88.6%	75.5%

However, we want to compare this results to the case where CNN is never deployed and we are using only first 1/3 of bike flows to see if our CNN over-performs the basic convergence in any way. This will be defined as a trivial case. In Table 16, this basic approach is evaluated. Note that 33.3% means getting 1 out of 3 links correct, 66.6% getting 2 out of 3, and 100% getting all top 3 most unbalanced links correct.

Table 16: Evaluation of the trivial approach

Method	**Jan 2019**	**Feb 2019**	**Mar 2019**	**Apr 2019**	**May 2019**
Trivial Top 3 Score	100%	33.3%	33.3%	33.3%	0%

Now, when comparing evaluation of CNN results to the trivial ones, it can be observed that there is a difference in months of January and April. In January, trivial approach over-performs pure CNN, while in April it under-performs. Although, at first glance, it might seems that the trivial approach is better, when using multivariate CNN RNN on top of CNN - we are able to get same result as trivial one for January. And in April, information lost with the trivial approach can not be compensated to beat the results got with CNN approach. So even though it is just one case, CNN does perform better in the long run, and intuition definitely tells us that with much more training and test data, we would be able to see many more examples of benefits. Not only that, but this thesis discussed only top 3 most unbalanced edges, and in case we were to use top 15 unbalanced edges, the results would be more detailed and diverse. But, because of the time restriction and difficulty to produce all the necessary matrices, this project focused solely on top 3 most unbalanced links.

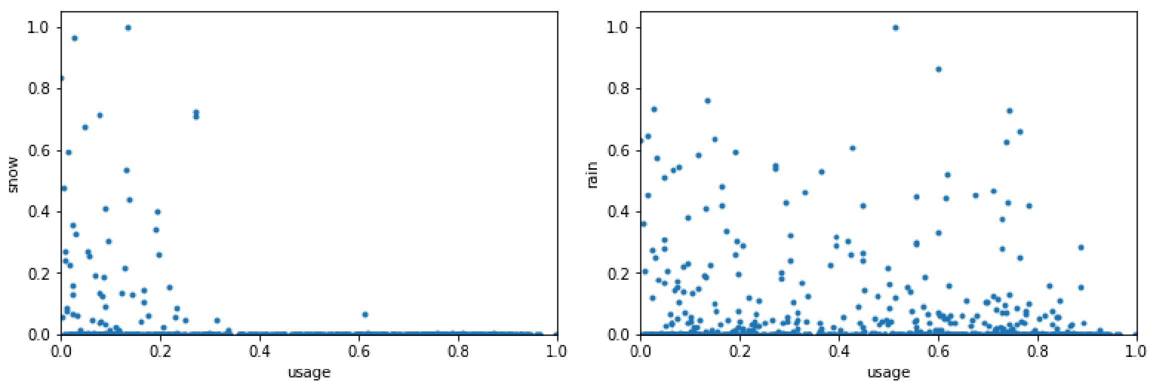


Figure 42: Correlation between bike usage and snow (left), bike usage and rain (right)

Another important insight comes from observing secondary weather dataset containing weather information. It came as a surprise to see that there is absolutely no correlation between bike usage and the intensity of snowfall or rainfall, as seen in Figure 42. This is quite counter intuitive as other bike sharing systems are quite dependant upon precipitation intensity. This finding can be maybe explained by a combination of BSS geographical positioning (state of Massachusetts) and the size of a bike sharing system itself (middle-sized). Additionally, this means that random forest prediction methods used for larger bicycle-sharing systems would not be as useful for the Boston BSS due to this unique weather property. More detailed investigation should be utilized on this matter, and finding another BSS with a similar property would be beneficial as well.

7.2 Conclusion

Applying a methodology described in this thesis, it is possible to use computationally lightweight combination of machine learning methods to grasp short-term approximation and prediction of spatial structures in the form of subgraphs, together with a more robust prediction of dynamic patterns and flows through the identified most unbalanced links. As a result, this method gives an opportunity for bike sharing companies to make a month long prediction on how many trucks for bike relocation will they need and in which streets or areas will they be most essential. The bottleneck of the project is its CNN segment due to inability to predict something that had never been observed before. Still, this thesis offers an approach that has not yet been considered. In addition, shortcomings of CNN are slightly alleviated using a subtle method of predicting monotonic properties, or trends of past values observed, which enables adding possible novel links in case their presence is making a detectable impact observed as a numeric quantity of their unbalanced flow.

As demonstrated in the discussion, we are able to predict most of the top unbalanced station tuples, and quantify the expected unbalanced flow between them with high accuracy. Of course, during more unstable months such as February and March, when a heavy snow season is expected in Boston, predicting spatial configuration can perform with lower accuracy than usual. This is, however, something that can not be mitigated in case of small tor medium BSS. In order for this project to be fully evaluated, the methodology presented in this thesis should be implemented within real life Boston Blue Bike relocation strategy, and achieved results compared to the same period in the past.

Moreover, the proposed methodology can be easily scaled up or scaled down, depending on the spatial granularity we would like to focus on (neighbourhood, town or municipality). Of course, whenever we are expanding or shrinking our spatio-temporal viewpoint, new set of stations will be inspected and thus, completely new matrices should be prepared accordingly. This is also a shortcoming, because as the number of relevant stations is growing, matrices expand as well. Not only that, but even in case when only one station should be added, the whole matrix must be re-arranged to mimic the station configuration in the real world. Luckily, any changes in the number of docks does not affect the model and that is why docks were not considered. This trade-off was necessary in order to ensure low memory and short model runtime requirements because short-term prediction should benefit with fast results, satisfactory prediction accuracy and model adaptability to be utilized for any BSS.

7.3 Future Work

There are many areas in the domain of BSS optimization and prediction that could be upgraded or further explored, including methods presented in this thesis. Building upon a proposed CNN architecture could be one of suggestions. Because the work presented here heavily relies on graph theory, one of the advanced GCNN models could be utilized. Moreover, when one has to deal with data defined on non-Euclidean domains, the definition of basic operations (such as convolution) becomes rather elusive. In that aspect, Geometric Deep Learning (GDL) area deals with the extension of ML techniques to graph and manifold structured data[22].

Another possibility could be to compare docked and dockless bike systems, and explore how the ML approaches in flow prediction coincide or differ one from another. Still, methods used would need to be adjusted to the fact that data available for both docked and dockless BSS is still very sparse, with an exception of bigger systems in China. Taking a whole another perspective, data scarcity could be mitigated and substituted with relevant secondary datasets. This additional datasets could introduce other modes of transportation (walking or vehicle based), or even socio-economical landscape and real estate market prices, which could vary depending on their proximity to mobility hubs.

One sharing system research area that had never been explored in depth before, and

[22]http://geometricdeeplearning.com/

just started flourishing a year ago is electric vehicle sharing systems study domain. This is an even greater challenge because, in addition to all the problems tackled in regular sharing systems, we also have to add electric power requirement variable and think about optimizing power distribution for such vehicles. Many new sharing transportation options are making advantage of this power source beside just bikes: scooters, mopeds, and even skateboards which are planned to be added in China. Having a variety of new ride sharing systems and many new companies growing their businesses across countries, research required in exploring this networks through the lens of ML and AI will be in demand now more than ever.

References

[1] P. DeMaio, "Bike-sharing: History, Impacts, Models of Provision, and Future," *Public Transportation, 12 (4): 41-56*, October 1, 2009.

[2] L. L. M. H. Schimmelpennink, "The Birth of Bike Share." October 1, 2012.

[3] D. Yanocha, "The bikesharing Planning Guide," *Institute for Transportation and Development Policy (ITDP)*, 2018.

[4] Z. Hong, A. Mittal, and H. S. Mahmassani, "Effect of Bicycle-Sharing on Public Transport Accessibility: Application to Chicago Divvy Bicycle-sharing System," *Transportation Research Board 95th Annual Meeting Transportation Research Board*, 2016.

[5] T. L. Susan Shaheen, "Reducing greenhouse emissions and fuel consumption: Sustainable approaches for surface transportation," *IATSS Res. 31,6-20*, 2007.

[6] Y. Zhang and Z. Mi, "Environmental benefits of bike sharing: A big data-based analysis," *Applied Energy, vol. 220, issue C, 296-301*, 2018.

[7] K. Sælensminde, "Cost-benefit analyses of walking and cycling track networks taking into account insecurity, health effects and external costs of motorized traffic," *Transportation Research Part A Policy and Practice 38(8):593-606*, October 2004.

[8] O. Pekka, S. Titze, A. Bauman, B. D. Geus, P. Krenn, B. Reger-Nash, and T. Kohlberger, "Health benefits of cycling: a systematic review." *Scand J Med Sci Sports.*, August 2011.

[9] R. J. Shephard, "Is active commuting the answer to population health?" *Sports Med.*, 2008.

[10] M. Peden, R. Scurfield, D. Sleet, D. Mohan, A. Hyder, E. Jarawan, and C. Mathers, "World report on road traffic injury prevention," *World Health Organization report, Geneva*, 2004.

[11] R. Beecham, J. Wood, and A. Bowerman, "Studying commuting behaviours using collaborative visual analytics," *Computers, Environment and Urban Systems Volume 47, Pages 5-15*, September 2014.

[12] D. Freund, S. G. Henderson, E. O'Mahony, and D. B. Shmoys, "Analytics and Bikes: Riding Tandem with Motivate to Improve Mobility," *Interfaces*, 2019.

[13] D. Chemla, F. Meunier, and R. W. Calvo, "Bike sharing systems: Solving the static rebalancing problem," *Discrete Optimization*, 2013.

[14] P.-C. Chen, H.-Y. Hsieh, X. K. Sigalingging, Y.-R. Chen, and J.-S. Leu, "Prediction of station level demand in a bike sharing system using recurrent neural networks," *IEEE 85th Vehicular Technology Conference*, 2017.

[15] R. C. Zheng, “Predicting bike sharing demand using recurrent neural networks,” *Procedia Computer Science 147:562-566*, 2019.

[16] L. Lin, Z. He, and S. Peeta, “Predicting station-level hourly demand in a large-scale bike-sharing network: A graph convolutional neural network approach,” *Transportation Research Part C: Emerging Technologies Volume 97*, December 2018.

[17] A. Yi, Z. Li, M. Gan, Y. Zhang, D. Yu, W. Chen, and Y. Ju, “A deep learning approach on short-term spatiotemporal distribution forecasting of dockless bike-sharing system,” *Neural Computing and Applications*, April 2018.

[18] G. McKenzie, “Docked vs. Dockless Bike-sharing: Contrasting Spatiotemporal Patterns,” *10th International Conference on Geographic Information Science*, 2018.

[19] D. O’Sullivan and D. Unwin, *Geographic Information Analysis*, 2002.

[20] F. Munoz-Mendez, K. Han, K. Klemmer, and S. Jarvis, “Community structures, interactions and dynamics in london’s bicycle sharing network,” *Proceeding UbiComp*, 2018.

[21] A. Sarkar, N. Lathia, and C. Mascolo, “Comparing Cities Cycling Patterns Using Online Shared Bicycle Maps,” *Transportation, Volume 42, Issue 4, pp 541559*, April 2015.

[22] J. Froehlich, J. Neumann, and N. Oliver, “Sensing and Predicting the Pulse of the City through Shared Bicycling,” *Proceedings of the 21st international jont conference on Artifical intelligence*, 2009.

[23] O. OBrien, J. Cheshire, and M. Batty, “Mining bicycle sharing data for generating insights into sustainable transport systems,” *Journal of Transport Geography 34*, 2014.

[24] M. Z. Austwick, O. OBrien, E. Strano, and M. Viana, “The structure of spatial networks and communities in bicycle sharing systems,” *PLoS ONE*, 2013.

[25] C. C. Robusto, “The cosine-haversine formula,” *The American Mathematical Monthly*, 1957.

[26] Z. Yang, J. Hu, Y. Shu, P. Cheng, J. Chen, and T. Moscibroda, “Mobility modeling and prediction in bike-sharing systems,” *PLoS ONE*, MobiSys ’16 Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services.

[27] Y. Pan, R. C. Zheng, J. Zhang, and X. Yao, “Predicting bike sharing demand using recurrent neural networks,” *Procedia Computer Science 147:562-566*, January 2019.

[28] S. Agatonovic-Kustrin and R. Beresford, “Basic concepts of artificial neural network (ann) modeling and its application in pharmaceutical research,” *Journal of Pharmaceutical and Biomedical Analysis*, 2000.

[29] L. van der Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of Machine Learning Research*, 2008.

[30] A. Tharwat, “Principal component analysis - a tutorial,” *IJAPR*, 2016.

[31] F. E. Pedroso, F. Angriman, A. L. Bellows, and K. Taylor, “Bicycle use and cyclist safety following boston’s bicycle infrastructure expansion, 2009-2012.” *American Journal of Public Health*, 2016.

[32] A. C. Lusk, P. G. Furth, P. Morency, L. F. Miranda-Moreno, W. C. Willett, and J. T. Dennerlein, “Risk of injury for bicycling on cycle tracks versus in the street,” *BMJ Publishing Group Ltd*, 2011.

[33] J. Garrard, S. Handy, and J. Dill, “Women and cycling,” *MIT Press*, 2012.

[34] K. von Lindenberg, “Comparative analysis of gps data,” *The Berkeley Electronic Press*, 2013.

[35] H. Coleman and K. Mizenko, “Pedestrian and bicyclist data analysis,” *NHTSAs Office of Behavioral Safety Research*, March 2018.

[36] A. F. Imani, N. Eluru, A. M. El-Geneidy, M. Rabbat, and U. Haq, “How land-use and urban form impact bicycle flows: evidence from the bicycle-sharing system (bixi) in montreal,” *Journal of Transport Geography*, December 2014.

[37] X. Wang, G. Lindsey, J. E. Schoner, and A. Harrison, “Modeling bike share station activity: Effects of nearby businesses and jobs on trips to and from stations,” *Journal of Urban Planning and Development*, March 2016.

[38] X. Zhou, “Understanding Spatiotemporal Patterns of Biking Behavior by Analyzing Massive Bike Sharing Data in Chicago,” *PLoS ONE*, October 7, 2015.

[39] B. Ratner, “The Correlation Coefficient,” *Journal of Targeting, Measurement and Analysis for Marketing*, May 18, 2009.

[40] G. P. Zhang, “Time series forecasting using a hybrid arima and neural network model,” *Elsevier Neurocomputing*, 2001.

[41] U. Pritzsche, “Benchmarking of classical and machine-learning algorithms (with special emphasis on bagging and boosting approaches) for time series forecasting,” *Ludwig Maximilian University of Munich*, 2015.

[42] A. A. Adebiyi, A. Adewumi, and C. Ayo, “Comparison of arima and artificial neural networks models for stock price prediction,” *Journal of Applied Mathematics*, March 2014.

[43] D. A. Dickey and W. A. Fuller, “Distribution of the estimators for autoregressive time series with a unit root,” *Journal of the American Statistical Association*, 2011.

[44] R. Mushtaq, “Augmented dickey fuller test,” *Social Science Research Network SSRN*, 2011.

[45] F. X. Diebold and L. Kilian, “Unit root tests are useful for selecting forecasting models,” *Journal of Business and Economic Statistics*, 2000.

[46] E. Zivot and J. Wang, “Rolling analysis of time series,” *Springer, New York, NY*, 2006.

[47] R. Adhikari and R. K. Agrawal, “An introductory study on time series modeling and forecasting,” *LAP Lambert Academic Publishing, Germany*, 2013.

[48] P. J. Brockwell and R. A. Davis, “Time series: Theory and methods,” *Springer, New York, NY*, 1991.

[49] E. P. George and G. M. J. , “Time series analysis: forecasting and control,” *Wiley*, 1970.

[50] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, , and S. Valaee, “Recent advances in recurrent neural networks,” *CoRR abs/1801.01078*, 2018.

[51] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks,” *Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA*, 2013.

[52] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, “How to construct deep recurrent neural networks,” *ICLR*, 2014.

[53] S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai, “Gradient descent finds global minima of deep neural networks,” *Proceedings of the 36 th International Conference on Machine Learning, Long Beach, California, PMLR 97*, 2019.

[54] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, “How to Construct Deep Recurrent Neural Networks,” *International Conference on Learning Representations*, 2014.

[55] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, 1997.

[56] C. Bergmeir and J. M. Benitez, “On the use of cross-validation for time series predictor evaluation,” *Information Sciences*, 2012.

[57] C. J. Willmott and K. Matsuura, “Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance,” *Inter-Research*, 2005.

[58] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016, http://www.deeplearningbook.org.

[59] Y. Guo, J. Zhou, Y. Wu, and Z. Li, "Identifying the factors affecting bike-sharing usage and degree of satisfaction in ningbo, china," *PLoS ONE 12(9): e0185100*, 2017.

[60] R. Uehara and Y. Uno, "Efficient algorithms for the longest path problem," *Algorithms and Computation, 15th International Symposium, ISAAC 2004, Hong Kong, China*, 2004.

[61] W. Zhao, "Research on the deep learning of the small sample data based on transfer learning," *AIP Conference Proceedings*, 2017.