Large-Scale Traffic Flow Prediction Using Deep Learning in the Context of Smart Mobility

by

Arief Koesdwiady

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2018

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

Supervisor(s):	Prof. Fakhri Karray Dept. of Electrical and Computer Engineering University of Waterloo
Internal Members:	Prof. Sherman Shen Dept. of Electrical and Computer Engineering University of Waterloo
	Prof. Liang-Liang Xie Dept. of Electrical and Computer Engineering University of Waterloo
Internal-External Member:	Prof. Kumaraswamy Ponnambalam Dept. of Systems Design Engineering University of Waterloo
External Examiner:	Prof. Karim Ismail Dept. of Civil and Environmental Engineering Carleton University

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Chapter 3 of this thesis consists of two articles that were co-authored by myself, my supervisor, and a post-doctoral fellow, Dr. Ridha Soua.

Chapter 4 has been incorporated within two papers that have been accepted and submitted for publication. The first paper was co-authored by myself, my supervisor, a Ph.D. student, Chaojie Ou, and a post-doctoral fellow, Dr. Safaa Al Bedawi. The second paper was co-authored by myself and my supervisor.

Chapter 5 of this thesis consists of an article that was co-authored by myself, my supervisor, and a Ph.D. student, Alaa El Khatib. This chapter also consists of an article that was co-authored by myself and my supervisor.

Abstract

Designing and developing a new generation of cities around the world (termed as smart cities) is fast becoming one of the ultimate solutions to overcome cities' problems such as population growth, pollution, energy crisis, and pressure demand on existing transportation infrastructure. One of the major aspects of a smart city is smart mobility. Smart mobility aims at improving transportation systems in several aspects: city logistics, info-mobility, and people mobility. The emergence of the Internet of Car (IoC) phenomenon alongside with the development of Intelligent Transportation Systems (ITSs) opens some opportunities in improving the traffic management systems and assisting the travelers and authorities in their decision-making process. However, this has given rise to the generation of huge amount of data originated from human-device and device-device interaction. This is an opportunity and a challenge, and smart mobility will not meet its full potential unless valuable insights are extracted from these big data.

Although the smart city environment and IoC allow for the generation and exchange of large amounts of data, there have not been yet well defined and mature approaches for mining this wealth of information to benefit the drivers and traffic authorities. The main reason is most likely related to fundamental challenges in dealing with big data of various types and uncertain frequency coming from diverse sources. Mainly, the issues of types of data and uncertainty analysis in the predictions are indicated as the most challenging areas of study that have not been tackled yet. Important issues such as the nature of the data, i.e., stationary or non-stationary, and the prediction tasks, i.e., short-term or long-term, should also be taken into consideration. Based on this observation, a data-driven traffic flow prediction framework within the context of big data environment is proposed in this thesis. The main goal of this framework is to enhance the quality of traffic flow predictions, which can be used to assist travelers and traffic authorities in the decision-making process (whether for travel or management purposes).

The proposed framework is focused around four main aspects that tackle major data-driven traffic flow prediction problems: the fusion of hard data for traffic flow prediction; the fusion of soft data for traffic flow prediction; prediction of non-stationary traffic flow; and prediction of multi-step traffic. All these aspects are investigated and formulated as computational based tools/algorithms/approaches adequately tailored to the nature of the data at hand. The first tool tackles the inherent big data problems and deals with the uncertainty in the prediction. It relies on the ability of deep learning approaches in handling huge amounts of data generated by a large-scale and complex transportation system with limited prior knowledge. Furthermore, motivated by the close correlation between road

traffic and weather conditions, a novel deep-learning-based approach that predicts traffic flow by fusing the traffic history and weather data is proposed.

The second tool fuses the streams of data (hard data) and event-based data (soft data) using Dempster Shafer Evidence Theory (DSET). One of the main features of the DSET is its ability to capture uncertainties in probabilities. Subsequently, an extension of DSET, namely Dempsters conditional rules for updating belief, is used to fuse traffic prediction beliefs coming from streams of data and event-based data sources.

The third tool consists of a method to detect non-stationarities in the traffic flow and an algorithm to perform online adaptations of the traffic prediction model. The proposed detection approach is developed by monitoring the evolution of the spectral contents of the traffic flow. Furthermore, the approach is specifically developed to work in conjunction with state-of-the-art machine learning methods such as Deep Neural Network (DNN). By combining the power of frequency domain features and the known generalization capability and scalability of DNN in handling real-world data, it is expected that high prediction performances can be achieved.

The last tool is developed to improve multi-step traffic flow prediction in the recursive and multi-output settings. In the recursive setting, an algorithm that augments the information about the current time-step is proposed. This algorithm is called Conditional Data as Demonstrator (C-DaD) and is an extension of an algorithm called Data as Demonstrator (DaD). Furthermore, in the multi-output setting, a novel approach of generating new history-future pairs of data that are aggregated with the original training data using Conditional Generative Adversarial Network (C-GAN) is developed.

To demonstrate the capabilities of the proposed approaches, a series of experiments using artificial and real-world data are conducted. Each of the proposed approaches is compared with the state-of-the-art or currently existing approaches.

Acknowledgments

All praise due to Allah for granting me the strength and knowledge to complete this work. I would like to express my sincere gratitude and appreciation to my supervisor Dr. Fakhri Karray for his continuous support, helpful discussion, and insightful guidance throughout my study at the University of Waterloo. I would like to express my gratitude to the members of my doctoral committee, Dr. Sherman Shen, Dr. Liang-Liang Xie, Dr. Kumaraswamy Ponnambalam, and Dr. Karim Ismail for investing time in reading and providing many valuable comments on my thesis.

Also, I would like to take time and thank my dear friends and colleagues at the Centre for Pattern Analysis and Machine Intelligence (CPAMI), whose motivational admiration, constructive comments, and influential criticisms, paved the way to accomplishing my goals. I am honoured to be a part of such an outstanding community.

Last and by far not least, I am indebted to my parents, brothers, and sisters for their continuous support, patience, and prayers. I owe a great deal of gratitude to my wife and my children for their continuous understanding, unlimited encouragement, and unending love during the years of my study.

Dedication

To Eyang and Moni,
my wife, Alvernia, and my children, Aidyn and Aatif,
my brothers and my sisters...

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average
BBA	Basic Belief Assignment
BoE	Body of Evidence
C-DaD	Conditional Data as Demonstrator
C-GAN	Conditional Generative Adversarial Network
CNN	Convolutional Neural Network
CRF	Conditional Random Field
DaD	Data as Demonstrator
DBN	Deep Belief Network
DL	Deep Learning
DNN	Deep Neural Network
DRC	Dempster's Rule of Combination
DSET	Dempster Shafer Evidence Theory
EKF	Extended Kalman Filter
EWMA	Exponentially-Weighted Moving Average
FHWA	Federal Highway Administration
FNM	Fuzzy-Neural Model
FoD	Frame of Discernment
GAN	Generative Adversarial Network

GMM	Gaussian Mixture Model
HCM 2010	Highway Capacity Manual 2010
ICT	Information and Communication Technology
IDM	Intelligent Driver Model
IoC	Internet of Car
IoT	Internet of Things
ITS	Intelligent Transportation System
KNN-NPR	K-Nearest Neighbour Non-Parametric Regression
LM	Levenberg-Marquardt
LWR	Lighthill-Whitham-Richards
MAE	Mean Absolute Error
MSTAR	Multivariate Spatial-Temporal Auto-Regressive
MTL	Multi-Task Learning
PCA	Principal Component Analysis
PCA-ALS	Principal Component Analysis with Alternating Least Square
PeMS	Performance Measurement System
RBF	Radial Basis Function
RBM	Restricted Bolztmann Machine
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SAFE	Spectral Analysis Feature Extraction
SARIMA	Seasonal-ARIMA
SMA	Simple Moving Average
SVM	Support Vector Machine
SVR	Support Vector Regression
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
VANET	Vehicular Adhoc Network

Chapter 1

Introduction

Presently, our cities suffer from ever increasing population growth and pollution leading to huge pressure on existing transportation infrastructure. The situation will be even more critical in the future. Recent statistics indicate that 60% of the worlds population will be living in cities by 2050 [6]. The smart city concept was proposed as a vialable recourse to deal with these issues. One of the main characteristics of smart cities is smart mobility. A smart-mobility-enabled city should be effortlessly accessible to its residents and visitors. Therefore, an efficient, intelligent, safe, comfortable, and multifaceted transportation system, which is linked to various Information and Communication Technology (ICT) infrastructures, has to be provided. With more than a billion cars on the roads today, which is expected to double to around 2.5 billion by 2050 [5], designing super-efficient navigation and safer travel journey are becoming a major challenge for transportation authorities. The development of Intelligent Transportation System (ITS) is a cornerstone in the design and implementation of smart cities. Indeed, building more roads will not radically solve the problem of large traffic congestion, fuel consumption, longer travel delays and safety.

Today, ITS is revolutionizing the how drivers and passengers use their transportation networks. The classical Vehicular Adhoc Networks (VANETs) were proposed to connect vehicles and infrastructure using Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communications, respectively. VANET plays an important role in safety-oriented applications, e.g., notifications of car accidents or hazardous weather condition, and traffic management applications. However, VANET constitutes only a small portion of a smart city. Moreover, some intrinsic characteristics of large cities hamper the efficient use of VANET such as tall buildings, complex road infrastructure and bad driver behavior. As such, smart cities should benefit from the Internet of Things (IoT) technology [165], clouds computing, and/or smart grids for the purposes of achieving more relevant, valuable and

anticipated decisions. Connected streets and cars represent a swarm of sensors that can feed citizens and authorities with rich data which can be used for predictive analytics. The target is to ensure smoother and sleeker traffic on the roads by anticipating traffic jams and roads congestion. In this regard, the IoT is paving the way for a new concept: the Internet of Cars (IoCs) [115]. Vehicles, drivers, "things" and environment are perceived as intelligent entities through the use of new computational paradigms such as deep learning, swarm computing and artificial intelligence [45].

Cisco estimates that 50 billion devices and objects will be connected to the Internet by 2020. Hence, connected cars evolve in a data-rich environment where they consistently generate and receive a variety of data that pours in from everywhere: weather stations, roads, traffic and social networks streams. Thus, this type of data has the great potential of enriching our experiences on how cities function and become a fuel that drives predictive related traffic application such as traffic flow prediction and informed decision-making.

The abstraction level of data ranges from raw low-level sensor data to higher-level information transmitted to connected cars and people for high quality predictive decision making. Thus, this unprecedented huge volume of data is becoming a major tool to help with traffic prediction and route planning. An interesting example is the city of Eindhoven where authorities have collaborated with IBM to design a traffic management solution that merges data coming from in-vehicles sensors with traffic gathered from roads [3]. Using Big data in IoC aims not only at storing and processing data but also at predicting and making decision in a timely and accurate way.

1.1 Motivation

Traffic flow prediction or forecasting is one of the major functionalities of ITS that should be achieved. It is particularly useful for planning proactive operations to alleviate road congestion. Moreover, timely and accurate prediction offers guidance to drivers so they can better plan their trip, choose the ideal departure time and best routes to avoid gridlock. However, traffic prediction is very challenging. Drivers use cities infrastructures to reach a specific destination following certain traffic rules. They interact with surrounding connected things, e.g., infrastructure, traffic light, cars, and with each other. During their trip, they are exposed also to changing weather conditions. Favorable or inclement weather will influence drastically drivers and their behavior on the roads.. Statistics made by the Federal Highway Administration (FHWA) reveal that 23% of road crashes are due to adverse weather conditions [2]. In given situation, it is crucial to investigate the impact of weather on traffic flow prediction. Quantifying such an impact will help transportation

operators and road users in terms of better pre-trip planning and best navigation strategies that match weather conditions.

In the context of smart cities, transportation networks are complex systems since they involve a considerable number of interconnected roads, which are often affected by several external factors: weather, scheduled and unscheduled events, and other traffic related conditions. Citizens also produce huge data using social media such as Twitter and Facebook that are installed on their mobile devices. Twitter is considered as a near real-time source of heterogeneous information of various importance. Twitter has over 300 million active users worldwide that generate 500 million tweets per day. In the case of traffic information, the users provide interesting and vital information such as travel plans or attendance of an event and share information regarding current traffic or road conditions. Indeed, social media contain unique information that can be used to extract traffic indicators. Thus, this type of data is enriching our knowledge of how cities function and is becoming a fuel that drives predictive related traffic application such as traffic flow prediction and informed decision making.

Traffic flow data are considered as time series data, which can be either stationary or non-stationary. Roughly speaking, a time series is considered as stationary if its statistical properties remain the same every time. The stationarity assumption is especially appealing in time series analysis due to the widely available models, prediction methods, and well-established theory. However, applying this assumption to real-world data such as traffic flow data, which mostly are non-stationary, might lead to inappropriate forecasts. The non-stationarity problem in traffic flow arises from the fact that when given the historical traffic information there are irregular traffic flows due to scheduled events such as constructions or unexpected traffic events such as accidents. In principle, due to unusual disruptions, long-term predictions may not be accurate enough for reliable practical use if we simplify the data to stationary. Non-stationarity, however, has been relatively scarcely studied and it is usually more challenging to handle.

Accurate and timely traffic flow prediction is essential for traffic management and allows travelers to make better-informed travel decisions. In some applications, it is often necessary to predict traffic flow not only accurately but also several steps ahead in the future. For example, in order for the traffic patch to manage a congested road and develop contingency plans, traffic dispatch may need to estimate traffic conditions at least 30 minutes in advance. However, most traffic flow prediction approaches were developed with single-step prediction methods. The multi-step prediction problem is significantly more difficult than its single-step variant and is known to suffer from degradation in predictions the farther we go in future time-steps. Therefore, it is essential to develop multi-step prediction approaches to achieve accurate multi-step traffic flow prediction.

Although the generation of huge, varied, dynamic, real-time, and interconnected data, which is known as big data, creates some challenges, it should be used to our advantage. The advantage for having access to big data comes with a number of issues that need to be dealt with:

i. How can these data be used to enhance the quality of traffic flow predictions?

ii. How can the inherent problem of big data be tackled during the predictions? This question can be elaborated as follows:

   - How to use the huge volume of data properly?
   - How to handle high velocity of data, which are being created in or near real-time?
   - How to fuse different type of data, hard and soft data, streams and event-based data?
   - How to select pertinent features or information for the prediction?

iii. How to tackle the non-stationarity aspect of traffic flow to maintain accurate prediction?

iv. How to achieve accurate multi-step traffic flow prediction horizon to have better-informed travel decisions?

1.2 Scope

Before narrowing the research scope, it is necessary to clarify the distinction between the definitions of several terms used in this research. *Hard data* are data that are generated by physical sensors, e.g., traffic flow, and weather data such as temperature, humidity, etc, whereas *soft data* encloses data that are generated by human operators, e.g., tweets and other social media posts. Based on how the data are generated, two types of data generation are distinguished: streams of data and event-based data. *Streams of data* are data that are generated continuously in sequence and are assumed to be available at each sampling time, whereas *event-based data* are data that correspond to relevant events and are not available at all sampling instances. Another important definitions are short-term and long-term traffic flow predictions. *Short-term* prediction is related to prediction tasks that involve short range of data from a few days to several months of data. *Long-term*

traffic flow prediction requires longer range data from several months to several years of data.

In general, time series can be categorized into two types: stationary and non-stationary. Roughly speaking, a time series is considered as stationary if its statistical properties remain the same every time. Formally, given a sequence $X_{t1}, \cdots X_{t_k}$ and a sequence $X_{t1+\tau}, \cdots X_{tk+\tau}$ in a time series, if the joint statistical distribution of the first sequence is identical as that of the second sequence for all τ, then the time series is *strictly stationary* [116]. This means that the moments, e.g., expectations, variance, third-order, and higher, are identical at all times. This definition is extremely strict for real-world applications. Therefore, a weaker definition, namely second-order or weak stationarity, is usually used to analyze time-series for practical applications. Second-order stationary time-series is a time series that has constant mean and variance over time. From this point, when a stationarity is mentioned, it means second-order stationarity.

This research mainly proposes a holistic framework that aims to address the traffic forecasting issues within the smart city context. Since a smart city environment is composed of data and information with big data properties, we tackle this problem from an artificial intelligence perspective. In the soft data part, the Twitter data annotation and extraction are not part of the current research since it is another big research area and requires intensive efforts. In this research, the area of interest is a specific area in California, i.e., Bay Area, District 4, San Francisco. This is due to the fact that we have easy access to this data, and the infrastructure is among the most complex a smart city could have.

1.3 Contributions

Although the smart city environment and IoC concept benefits from a variety of sources generating data and information, there has not been any well-defined and concrete approaches that take advantage of this diversity in a way to assist the travelers and traffic authorities. The main reason for this scarcity is most likely related to the fundamental challenges of the big data problem that needs thorough consideration. Mainly, the issues of types of data and uncertainty analysis in the predictions are indicated as the most challenging areas of study that have not been tackled in the past. Based on this observation, a traffic flow prediction model using big data is proposed to enhance the quality of the traffic flow prediction, which can be used to assist travelers and traffic authorities in the decision-making process. Our work is centered around coming up with adequate tools to deal with the huge amount of data available to us, which comes from different sources, under different bandwidth and under different formats.

1.3.1 Large-scale Traffic Flow Prediction Using Hard Data

This work proposes a model that is capable of tackling the inherent big data problems and dealing with the uncertainty in the predictions. It relies on Deep Belief Networks (DBNs) introduced by Hinton in [67], as the main approach underpinning the traffic flow prediction process. The main reason for choosing DBN is its capability in handling huge amounts of data generated by a large-scale and complex transportation system with limited prior knowledge. DBNs have been found to be powerful in learning the representative features from the data. Furthermore, a decision-level data fusion scheme is introduced in this work. This scheme fuses streams of data coming from the road sensors and weather stations.

1.3.2 Large-scale Traffic Flow Prediction Using Soft Data

The second scheme fuses the streams of data and event-based data using Dempster Shafer Evidence Theory (DSET) [135]. One of the main features of DSET is its ability to capture the uncertainty in the probability. The second scheme of fusion is done at the decision level after the prediction values are transformed into hard clusters. Since the predictions contain prediction errors, the DSET is able to capture this error as uncertainty in the conditional probability of the clusters. Moreover, since the data coming from the social media may not be completely reliable, an extension to classical DSET, namely conditional updating rule of DSET, is used. In conclusion, since the proposed model is developed within the context of the smart city, it is capable of exploiting various technologies and data available in this environment to offer efficient mobility within limited infrastructure capacity.

1.3.3 Non-Stationary Traffic Flow Prediction

In the long-term traffic flow prediction, the stationarity assumption may not reflect realities and may lead to incorrect predictions. A method to detect nonstationarities in the traffic flow and update the predictor accordingly is proposed. The proposed detection mechanism is developed by monitoring the evolution of the spectral contents of time series through a distance function. This method is designed to work in combination with state-of-the-art machine learning methods in real time by informing the online predictors to perform necessary adaptation when a non-stationarity presents. Furthermore, an algorithm to proportionally include some past data in the adaption process to overcome the *Catastrophic Forgetting*[1] problem is also proposed. The proposed method is suitable for big

[1]The problem of forgetting the previously learned data when learning new data [49]

data environment since it minimizes the computational resources needed when updating the predictor.

1.3.4 Multi-step Traffic Flow Prediction

Multi-step time series prediction is known to suffer from increasing performance degradation the farther in the future the predictions are made. In this work, two approaches to address this weakness in recursive and multi-output prediction models are introduced. In particular, a model that allows recursive prediction approaches to take into account the time-step index when making predictions is developed. In addition, a conditional generative adversarial network-based data augmentation model to improve prediction performance in multi-output models is proposed. In order to evaluate the proposed approach, several experiments involving previous strategies to handle multi-step time series prediction are conducted.

1.4 Thesis Outline

This thesis is composed of the following chapters:
Chapter 1 is an introduction that presents the motivation behind the research work as well as contributions made in the area.

Chapter 2 provides a literature review of various work related to the research.

Chapter 3 presents methods to predict traffic flow using hard and soft data. It discusses the details of the proposed approach and the benefits of using the combination of hard and soft data

Chapter 4 presents an approach to handle nonstationarity in traffic flow prediction. A nonstationarity detection module and a mechanism to embed the detection module into predictors are introduced in this chapter.

Chapter 5 presents approaches to improve multi-step traffic flow prediction. A modified recursive method and a data augmentation method using Generative Adversarial Networks (GANs) are presented in this chapter.

Chapter 6 provides a summary of the contributions of this research work and discusses possible areas for future research.

Chapter 2

Background and Related Work

This chapter provides technical background used in this thesis and reviews the related work in the literature by first emphasizing the importance of the concept of smart mobility in smart city, and then introducing recent and state of the art modeling approaches for traffic flow prediction. Accordingly, section 2.1 presents the concept of smart mobility in smart cities perspective. Section 2.2 provides an overview on traffic flow prediction, which is followed by the review on research work tackling the traffic flow prediction major issues in section 2.3. Section 2.4 presents technical concepts on deep learning as the main predictor used in this work. Section 2.5 presents the technical background of Dempster Shafer Evidence Theory (DSET) data fusion and its extension as well as literature review of DSET data fusion.

2.1 Smart Mobility in Smart City

In this section, the concept of smart city is briefly highlighted in terms of its features and application domains. A smart city needs an integrated system to collect data from various heterogeneous sources such as road sensors, weather stations, security cameras, internet, etc [117]. Furthermore, many smart cities are designed and developed on the basis of sophisticated and intelligent systems that can sense, learn, understand, and act towards their environment [23, 104]. In this type of system, a considerable number of real-time information is processed and analyzed to enhance and/or optimize the performance of the systems and assist authorities in the planning, decision-making, and control activities. In the specific case of traffic flow prediction, a combination of the right data and of the right policies can make traffic flow run much more smoothly.

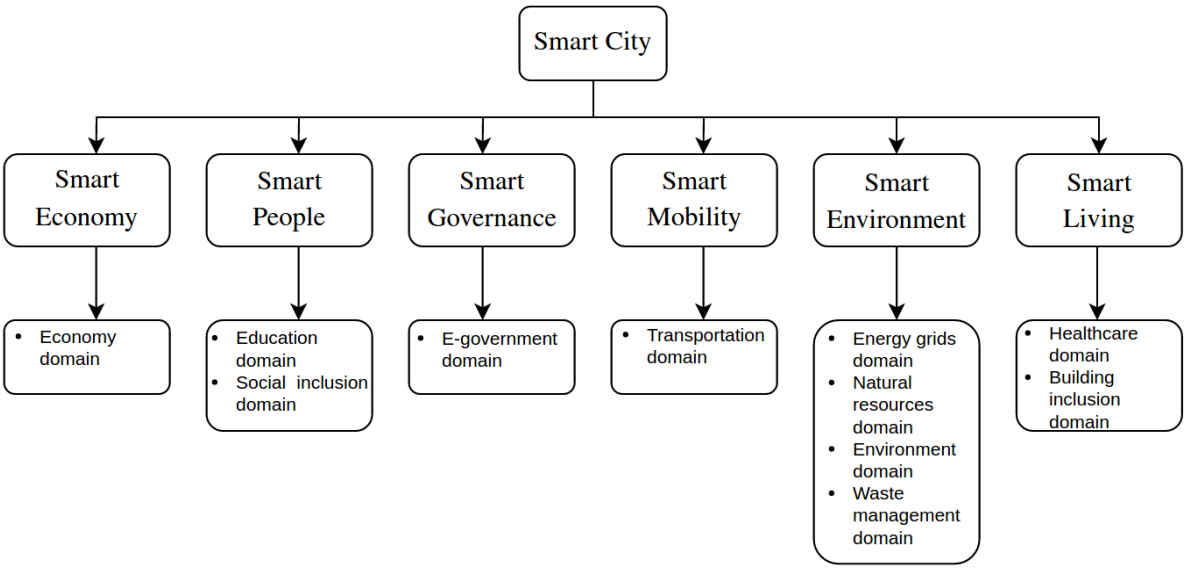


Figure 2.1: Taxonomy of Smart City.

The domains in which smart city concept is developed can be categorized as *hard* and *soft* domains [117]. *Hard* domains consist of eight sub-domains: energy grids, natural resources, environment, transportation, buildings, waste management, healthcare and public security, whereas *soft* domains consist of education, social inclusion, e-government, and economy. In the hard domain settings, the deployment of Information and Communication Technology (ICT) is crucial; the use of sensors, wireless technologies, and software solutions to handle big data and assist a city to sense and act can be most applicable in this domain. According to [51], the two domains of applications can be further grouped into six characteristics of smart city. The complete taxonomy of smart city is depicted in Fig. 2.1

The transportation domain, also known as the smart-mobility aspect of smart city, has several main objectives: to optimize logistics and transportation based on traffic conditions and energy consumption [14, 110, 51]; to provide travelers and authorities with dynamic and multi-modal information for transport efficiency [35, 156]; and to assure sustainable public transportation [25, 87, 114, 145]. Furthermore, the smart mobility aspect can be classified into three sub-aspects [117]:

- City logistics–Improving logistics flows in cities by effectively integrating business needs with traffic conditions, geographical, and environmental issues.

- Info-mobility–Distributing and using selected dynamic and multi-modal information, both pre-trip and, more importantly, on-trip, with the aim of improving traffic and transport efficiency as well as assuring a high quality travel experience.
- People-mobility–Innovating the mobility of people in cities, such as the development of public transport modes and vehicles that are supported by advanced technologies and proactive citizens' behaviors.

From the mentioned sub-aspects, it can be concluded that multi-modal traffic flow predictions play an important role in realizing smart mobility. The necessity of using road traffic monitoring technology that links data coming from people, cars, and sensors to provide predictive tools for a smarter motorized society is increasing. Specifically, data-driven prediction techniques are among those methods that due to the characteristic of smart city and Internet of Car (IoC). However, designing a generic model that utilizes a significant portion of such broad selection of data resources, and automatically fuses the information of interest while handling the big data problem, is still an open problem that requires more in-depth investigation. The next section of this chapter reviews the contemporary research on traffic flow prediction.

2.2 Traffic Flow Prediction: An Overview

Traffic flow prediction has been long considered as an important problem for intelligent transportation systems. There are three parameters representing traffic flow: rates of flow, speed, and density [146]. These elements can be used to infer road conditions such as traffic breakdown, types of congestion, and the capacity drop after a breakdown. The traffic flow is usually represented in the form of time series of the aggregated quantities, where the time scale ranges from minutes to a few hours. However, to maintain the sustainability of the measured volume, the Highway Capacity Manual 2010 (HCM 2010) recommends using 15-minute intervals [101].

Two sources of data are usually used in traffic flow prediction: road sensors and entrance and exit detectors. The road sensors are inductive loop detectors located on each road. These detectors provide a direct measurement of volume as well as of speed when they are used in pairs. Furthermore, the measured volume can be used for traffic flow prediction task. In the second source, the data can be used to monitor when a car is entering and exiting a road. This data can be used to forecast the traffic flow in exit station. In addition to the two common sources of data, a various data sources such as those obtained

from surveillance cameras, radar, and LiDAR, have been used in traffic flow prediction research [167, 62, 153].

In this thesis, the data are collected from the California Freeway Performance Measurement System (PeMS) using inductive loop detectors[1]. To illustrate the traffic prediction data arrangements, consider the traffic flow of the i-th observation point, i.e., a road or a station, at the t-th time point denoted by $f_i(t)$. At instant time T, the main task is to predict traffic flow in k_1-step time in the future $f_i(T + k_1)$ using the traffic flow sequence $F = \{f_i(t)|i \in O, t = T, T-1, T-2, ..., T-k_2\}$, where k_2 is the time step in the past, and O is the complete set of all observation points. In other words, the traffic flow prediction at point i is performed based on present and historical data from all observation points.

2.3 Traffic Flow Prediction in the Literature

In this section, a taxonomy of traffic flow prediction approaches is introduced. This taxonomy is developed based on the types of information and models used for predictions and is depicted in Figure 2.2. Generally, traffic flow prediction models are developed using two approaches: analytical and data-driven approaches. In the analytical approach, traffic flow prediction models are developed using known accurate physical laws and can be categorized into three types according to the granularity of the information used: microscopic, macroscopic, and mesoscopic. In the data-driven approach, prediction models are developed using available input-output data from real systems that are used to fit specific mathematical structures. Furthermore, this approach can be further classified into three types based on the types of models: parametric, non-parametric, and hybrid models. Additionally, non-stationary and multi-step traffic flow predictions are discussed in the non-parametric approach since these topics are also major parts of the thesis. Subsequently, the following subsections present the literature review of analytical and data-driven approaches for traffic flow predictions.

2.3.1 Analytical Approach of Traffic Flow Modelling

The analytical approaches can be further classified into three levels of perspectives: microscopic, mesoscopic, and macroscopic perspectives. The microscopic perspective illustrates the most basic behavior of individual vehicles such as the position and velocity. In contrast, the macroscopic perspective describes the traffic flow as a stream of vehicles that has

[1] http://pems.dot.ca.gov

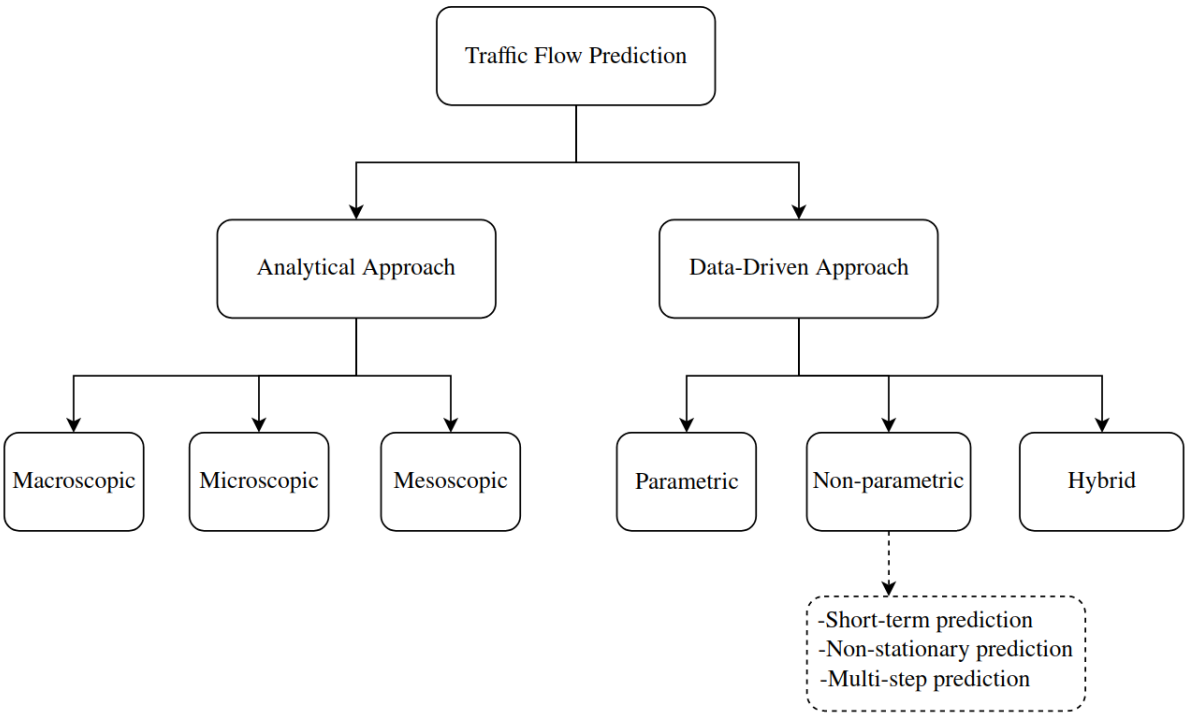


Figure 2.2: Taxonomy of traffic flow prediction.

characteristics like flow, density, speed, etc. Lastly, the mesoscopic perspective combines the individual behavior of the vehicles and macroscopic properties of the traffic flow. In this subsection, the three perspectives of analytical modeling approaches are investigated.

Microscopic perspective

In the microscopic perspective, the behaviour of each individual vehicle is modeled either using commonly used car-following models or most recently developed cellular-automata models. There are three major types of car-following models in the literature: safe-distance, stimulus-response, and action-point models.

In the earliest safe-distance car-following models, the position of the leader is expressed as a function of the position of the follower [121]. In [52], safe-distance car-following models are refined by the assumption that the speed of the vehicles is within the safety requirements. Later, a simple safe-distance model with delay is introduced in [118, 119]. Using Lagrangian coordinate system, it is shown in [90] that the model in [119] is equivalent

to a macroscopic model. Finally, an extension to the model in [119] is proposed in [88].

Another type of car-following models is stimulus-response models, which are developed based on the assumption that drivers accelerate according to three stimuli: own current velocity, spacing with respect to the leader vehicle, and relative velocity with respect to the leader vehicle. The most notable model of stimulus-response models is Gazis-Herman-Rothery model [50]. Later, the optimal velocity model is developed in [18] and extended by introducing delay in [17]. Recently, an Intelligent Driver Model (IDM) based on acceleration and velocity of the vehicles is proposed in [147].

The last type of car-following models is action-point models. The main contribution of the action point models is that the models include the fact that at the large freeways the driving behaviour is not affected by the behaviour of other vehicles [24]. The main difference between cellular-automata and car-following models is that the space in the cellular-automata models is discretized. The cells, which are partitions of roads, may or may not have vehicles inside them. The cellular-automata models are introduced in [111, 81].

Macroscopic perspective

There are two majors types of macroscopic models: kinematic wave model and higher-order model. The kinematic wave model is introduced in [94, 130] and known as Lighthill-Whitham-Richards (LWR) model. In this model, the dynamics of the traffic is represented by a partial differential equation that models the conservation of vehicles. The model is developed based on the assumption that the vehicles reach the new equilibrium velocity immediately after a change in the traffic state, which implies infinite acceleration. An extension of the models is proposed in [89, 91]. It involves bounded-acceleration in the dynamics. In [75], lane changing in a discretized version of the LWR model is included to eliminate the breakdown problem. The stochastic version of the kinematic wave model is proposed in [70]. This version uses breakdown probabilities to capture the notion that the break down takes place in different densities. Recently, a multi-class model based on the kinematic wave traffic flow model is introduced in [149, 150].

The higher-order models introduce velocity dynamics,i.e., acceleration, described by a fundamental relation. In the early development of the models, the higher-order models received a lot of critiques because they are not anisotropic. These critiques have motivated several researchers to deal with the problem. The most well-known model addressing the anisotropic issue is introduced in [15], and the multi-class version of the model is proposed in [16].

Mesoscopic perspective

Mesoscopic traffic flow models consider individual vehicles in aggregate terms such as probability distributions. The most popular kind of mesoscopic models is gas-kinetic models. These models describe vehicles as the dynamics of velocity distribution functions. An improvement of gas-kinetic models is motivated by the development of a multi-lane version of Paveri-Fontanas model, [65]. In [69], a generic gas-kinetic traffic flow model including the previously mentioned model[65] is introduced. Finally, a gas-kinetic model that includes a simple car-following model is introduced in [143].

2.3.2 Data-driven Approach of Traffic Flow Modeling

Significant efforts have been made in recent years to address the problem of data-driven traffic flow prediction [96]. In general, data-driven traffic flow prediction approaches can be categorized into three categories: parametric, non-parametric, and hybrid.

Parametric Approach

In the parametric approach, the structure of the model is predetermined; hence, only parameters of the model are estimated to fit the data. The well-known auto-regressive model with its extension such as Autoregressive Integrated Moving Average (ARIMA) is investigated in [157]. To optimize the ARIMA traffic flow model, a heuristic approach is introduced in [144]. Often time series possess a seasonal component that repeats every certain period of time. In order to deal with seasonality in the traffic flow data, a Seasonal-ARIMA (SARIMA) model is introduced in [31]. To include the transient behaviour of the traffic network, a Multivariate Spatial-Temporal Auto-Regressive (MSTAR) model is introduced in [106]. In [59], an Autoregressive Moving Average (ARMA) model with generalized auto-regressive conditional heteroscedasticity structure is proposed. This structured is processed online using layered Kalman Filter. [73]. To overcome the linearity assumption in the ARMA model, a non-linear kernel-based stochastic time series method with state space reconstruction capability using Extended Kalman Filter (EKF) is proposed in [19]. A prediction mobile that combines a Kalman Filter and Gaussian Mixture Model (GMM) is introduced in [74].

Despite the encouraging results reported in the mentioned studies, the linear structured assumption of the time series approaches presents an inconvenience for traffic prediction. Traffic flow is stochastic and nonlinear in nature. In addition, roads are inter-connected;

hence, it is very difficult for the time series approach to capture this behavior. Since transportation networks are complex and very correlated, it is crucial to predict traffic flow from a network perspective. Finally, time series based approaches are more prone to large errors in complex traffic forecasting.

Non-parametric Approach

To overcome the issues encountered in the parametric approaches, researchers have proposed non-parametric solutions. The term non-parametric does not imply without parameters, but it is indicative of the fact that the number and nature of the parameters are flexible and not fixed in advance. Several researchers have proposed non-parametric regression methods to solve traffic prediction problem. In [32], a K-Nearest Neighbour Non-Parametric Regression (KNN-NPR) method is used to predict traffic flow using real-world historical data. An online boosting regression technique that is activated when predicting traffic under abnormal conditions is proposed in [159].

In [161], a Fuzzy-Neural Model (FNM) with two modules is proposed to predict traffic flows in an urban street network. The first module, i.e., fuzzy module, clusters the input of the system that is used in the second module, i.e., Artificial Neural Network (ANN) module. A novel ANN training method is introduced in [30] to enhance the accuracy of the previously developed ANN training methods for traffic flow prediction; this novel method uses the hybrid exponential smoothing method and the Levenberg-Marquardt (LM) algorithm. In [128, 136], ANN algorithms based on niche genetic algorithm and ant colony optimization are investigated respectively.

Support Vector Machine (SVM) is a powerful classification method and has been proven to out-perform other classification methods in a number of large dimensional data set. The regression version of SVM is known as Support Vector Regression (SVR). In [71], an SVR for traffic flow prediction is proposed. The model uses particle swarm optimization algorithm to optimize the parameters of the SVR model. One of the limitations of SVR lies in the choice of the suitable kernel function. In [155], the use of Chaos-Wavelet Analysis-SVM to construct the kernel function is proposed. Traffic flow prediction in large networks using SVR is proposed in [13]. The method uses unsupervised-learning methods to mine spatio-temporal performance trends at the network level. For comparison, table 2.1 depicts the advantages and challenges associated with the parametric, ANN, and SVR models [134].

Recently, specific attention was given to Deep Learning (DL) [99, 72] to forecast traffic flow. This is motivated by the fact that other techniques either require a prior knowledge

Table 2.1: Advantages and challenges of parametric, ANN, and SVR models

Prediction Method	Advantages	Challenges
Auto-regressive and Kalman Filter	- Suitable for low order model	- Based on linear model assumption
	- Guaranteed to converge	- High computation for high-order models
	- The model directly measures the Mean-Squared Error (MSE)	
ANN	- Not model dependent - Independent on linear, stationary process - Feed forward efficient computation	- Number of parameters is large - Free parameters are selected experimentally - Expensive computation on the training process - Not guaranteed to converged to the global optima
SVR	- Not model dependent	- Free parameters are selected experimentally
	- Independent on linear, stationary process	- Expensive computation on the training process
	- Guaranteed to converge to the optimal solution	- Kernel selection can be difficult

of specific domains for feature extraction and are shallow in architecture. The existing neural-based approaches use a limited number of layers, i.e., one layer. For these shallow structures, it is easy to overfit the data once the number of layers is increased. Deep learning, on the other hand, could learn features with a less prior knowledge, and also has been reported to out-perform other learning methods in object, image and speech recognition tasks [112, 93, 108]. In addition, deep learning is able to handle a large amount of data and high dimension of features that usually characterize transportation systems in a smart city.

In [99, 72], the authors did not take into account factors that are affecting the traffic flow such as weather, which has a significant impact on traffic flow [152, 148]. Although in [99] the authors mentioned the big data aspects, the paper did not cover all the big data components. Moreover, this paper considers only traffic flow during the working days,

while the traffic flow on the weekends has distinctive patterns. The concept of big data is also covered in [98]; however, the incorporation of the smart city concept, which generates enormous and different kinds of data, is not presented.

It has been shown that weather has a significant influence on traffic flow prediction [127]. In [41], authors proposed a neuro-wavelet based framework to forecast hourly traffic taking into account the effect of rainfall. However, the dataset used in this work is limited to only two traffic junctions in Dublin city. Moreover, only weekdays of the two first weeks of January were considered for traffic prediction. In [169], authors investigated the impact of snowy and icy conditions on traffic with data from Buffalo, New York. They modeled the impact of adverse weather conditions on traffic speed. In this work, the authors used a simple linear regression method which assumes linearity in the structure of the model with separate learning. Since transportation networks are complex and very correlated, it is crucial to predict traffic flow from a network perspective. Thus, parametric approaches are more prone to large errors in traffic forecasting.

Social media information can enhance traffic prediction accuracy. Relevant information about city events that may acutely impact the driver behavior can be extracted from Twitter or Facebook posts. The study proposed in [63] use Twitter data as an external data source for improving long-term traffic prediction in San Francisco Bay area. They also analyzed the correlation between road traffic indicators and tweets counts. Finally, traffic indicators extracted from tweets are incorporated into the linear regression model for traffic forecasting beyond one hour. However, their auto-regression model that incorporates traffic and social activity intensity is not well adapted to the complexity of high dimension of features characterizing transportation systems. Moreover, the distinction of events extracted from Twitter data is neglected. The authors of [95] have improved the study proposed in [169] by including weather indicator extracted from Twitter data. The authors apply linear regression model using Twitter-based weather indicators. This study shares with other mentioned work the limitations of linear regression models. In addition, the authors extracted only weather-related information from Twitter data. However, congestion-causing events such roadway construction and maintenance; planned special event; traffic incident; and emergency road work disrupt the normal flow of traffic and result in reduction in roadway capacity.

In the long-term traffic flow prediction, the stationarity assumption is not appealing. Non-stationary time series prediction, in general, is a challenging problem that researchers have been trying to solve. One of the solutions to handle non-stationarity is to consider non-stationary time series as a collection of piece-wise, or locally, stationary time-series. This means the parameters of the time series are changing but remain constant for a specific period of time. In relation to prediction problems, using the piece-wise stationarity

assumption, the stationary part of the time series can be learned using stationary methods. Subsequently, the predictor is updated when the time-series move to a different stationary state. Therefore, it is imperative to continuously check whether the time series is stationary or not.

A vast array of tools for detecting non-stationarity has been introduced by researchers. Most of the detection mechanisms are based on spectral densities [123, 7], covariance structures comparisons [12, 20], and, more recently, using locally stationary wavelet model [34, 85]. These tests are based on a specific choice of segments of the data, which is sometimes delicate and highly subjective. In [42], the test is based on the discrete Fourier transform using the entire length of the data, which is undesirable in online settings.

In the machine learning community, researchers are more interested in concept drift detection since most of them are dealing with classification problems [46, 131, 54]. However, in regression problems such as time-series predictions, it is more suitable to consider non-stationarity. Non-stationarity is a more general concept in a sense that time-series without concept drift might contain non-stationarity, e.g., a near-unit-root auto-regressive process. Although the concept is not drifting, i.e., the parameters of the model are static, the time series evolution might contain changing mean and variance.

In spite of all these efforts, there has been limited amount of work to tackle non-stationary traffic flow prediction. In [125], the authors developed a two-step approach based on the stochastic differential equations to improve short-term prediction and used a first order local polynomial as drift term for each residual in the non-stationary time series. However, this work used very limited datasets; as an example, the PeMS dataset [2] collected in the period of October 1, 2009 to November 30, 2009. This dataset is not sufficient to capture the long-term stationarity of the traffic flow. In order to validate the performance of a predictor under non-stationary conditions, it is necessary to test the predictor with long-term traffic flow data. In [154], traffic flow prediction under non-stationary conditions is modeled based on ensemble real-time sequential extreme learning machine. The proposed method quickly trains the data and incrementally update the model when the new data arrive. Indeed, blindly updating the model as time progresses is computationally expensive.

In its simplest form, traffic flow prediction problem is restricted to predicting a single time-step into the future. Multi-step traffic flow prediction extends this set-up to the case where predicting multiple time-steps into the future based on some finite history is of interest. This problem is significantly more difficult than its single-step variant and is known to suffer from degradation in predictions the farther we go in future time-steps. Nonetheless, we are often interested in forecasting more than a single time-step ahead. In [120], four

[2] California Performance Measurement System, `http://pems.eecs.berkeley.edu`

adaptive Kalman filter approaches for the multi-step ahead forecasting of short-term traffic flow is proposed, which were developed using recursive strategy. Furthermore, a spatio-temporal recurrent convolutional network for traffic forecasting is introduced in [164]. The proposed SRCN exploits the advantages of deep convolutional neural networks and long short-term memory neural networks. In [163], a short-term traffic prediction model based on the k-nearest neighbor algorithm is proposed. The approach uses the time-varying and continuous characteristic of traffic flow and implements the single-time-step model for the multi-time-step prediction. All of these approaches do not explicitly develop ways to tackle the multi-step prediction. Therefore, the degradation of the performance in the far future is still a challenge.

Hybrid Approach

To enhance the performance and to find the suitable parameters of the existing models, several works have investigated hybrid approaches. In [105], a combination of analytical approaches, i.e., LWR model, and knowledge-based models, i.e., Bayesian, ARIMA, for traffic flow prediction is investigated. A linear combination of ANN and Radial Basis Function (RBF) networks according to a heuristic credit assignment algorithm based on Bayes' rule is proposed in [170]. Another hybrid model is presented in [168]. In this model, the traffic flow is predicted by decomposing the data into three components: a periodic trend extracted using the spectral analysis technique, a deterministic part introduced by the ARIMA model, and the volatility estimated by the GJR-GARCH model. By combining several techniques, these approaches are more adaptive. Although the above mentioned hybrid models are flexible, they still do not capture the notions of big data and fully take profit from spatial information collected from the whole road network. Moreover, these studies rely on only information collected by sensors such as GPS, loop detectors and smartphones.

2.4 Deep Learning: An Overview

Deep learning refers to models that are constructed by multiple processing layers to learn data representations with multiple level of abstraction [92]. The learning process can be done in supervised, semi-supervised, or unsupervised way. Furthermore, deep learning has been successfully applied and has achieved state-of-the-art performances in different domains such as image recognition [64], speech recognition [66], machine translations [138], and genomics [8]. Deep learning has been developed with different architectures

such as deep Convolutional Neural Network (CNN)) [86], deep Recurrent Neural Network (RNN) [56], Deep Belief Network (DBN) [67], and Deep Neural Network (DNN). This thesis implements DBN and DNN as the main models for traffic flow predictions. Next, the technical details of DBN and DNN are presented.

2.4.1 Deep Belief Networks (DBNs)

DBN is suitable for dealing with huge amount of data generated by complex interconnected systems. It is formed by stacking Restricted Bolztmann Machines (RBMs), which are then trained greedily using unsupervised algorithms [67]. An RBM is an undirected graphical model that has no connections within visible ($\mathbf{x}$) or hidden ($\mathbf{h}$) units, as illustrated in Figure 2.3.

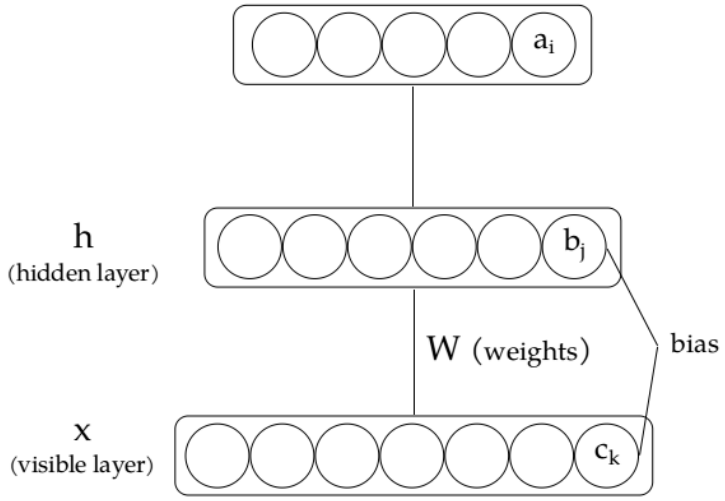


Figure 2.3: Restricted Boltzmann Machines

RBMs are developed based on the inter-layer interaction energy and bias energy of each layer. Mathematically, the energy function of an RBM is defined as:

$$E(\mathbf{x}, \mathbf{h}) = -\mathbf{b}^\top \mathbf{x} - \mathbf{c}^\top \mathbf{h} - \mathbf{h}^\top \mathbf{W} \mathbf{x} \tag{2.1}$$

where $\mathbf{b}$ and $\mathbf{c}$ are bias vectors associated with the hidden and visible layers respectively, and $\mathbf{W}$ is the weights between the visible and hidden layers. We denote the set of parameters $\mathbf{b}, \mathbf{c},$ and $\mathbf{W}$ by θ. An RBM defines a distribution, which involves the hidden units,

over the visible units. The distribution of observing a particular visible and hidden units configuration based on the energy function is given by:

$$P(\mathbf{x}, \mathbf{h}) = \frac{e^{-E(\mathbf{x},\mathbf{h})}}{Z} \tag{2.2}$$

where Z is the partition function. Using Equation 2.2, the distribution of observing a set of visible units is defined as:

$$P(\mathbf{x}) = \sum_{\mathbf{H}} P(\mathbf{x}, \mathbf{h}) = \exp \frac{-F(\mathbf{x})}{Z} \tag{2.3}$$

where $F(X)$ is known as the free energy term, and is defined as follows:

$$F(\mathbf{x}) = \exp\left(\mathbf{c}^\top \mathbf{x} + \sum_{j=1}^{H} \log(1 + \exp(b_j + \mathbf{W}_{(j,:)}\mathbf{x}))\right)/Z \tag{2.4}$$

To train an RBM, the average negative log-likelihood function of all training points $(t = 1, ..., T)$ needs to be minimized. The function is set as follows:

$$NLL = \frac{1}{T}\sum_{t} l(f(\mathbf{x}^{(t)})) = \frac{1}{T}\sum_{t} -\log P(\mathbf{x}^{(t)}) \tag{2.5}$$

Then, the function is optimized with respect to θ using stochastic gradient descent algorithm; the result of the optimization is defined as:

$$\begin{aligned}\frac{\partial - \log P(\mathbf{x}^{(t)})}{\partial \theta} &= \mathbb{E}_{\mathbf{h}}\left[\frac{\partial E(\mathbf{x}^{(t)}, \mathbf{h})}{\partial \theta} \,\middle|\, \mathbf{x}^{(t)}\right] \\ &\quad - \mathbb{E}_{\mathbf{x},\mathbf{h}}\left[\frac{\partial E(\mathbf{x}, \mathbf{h})}{\partial \theta}\right]\end{aligned} \tag{2.6}$$

The second term on the right-hand side of the equation is hard to compute since we have to make an exponential sum over both $\mathbf{x}$ and $\mathbf{h}$. To address this problem, the authors of [26] proposed the contrastive divergence algorithm, which has three main steps:

1. Estimate a point $\tilde{\mathbf{x}}$ to replace the expectation.
2. Estimate the point using Gibbs sampling.
3. Start the sampling chain at each observation $\mathbf{x}^{(t)}$.

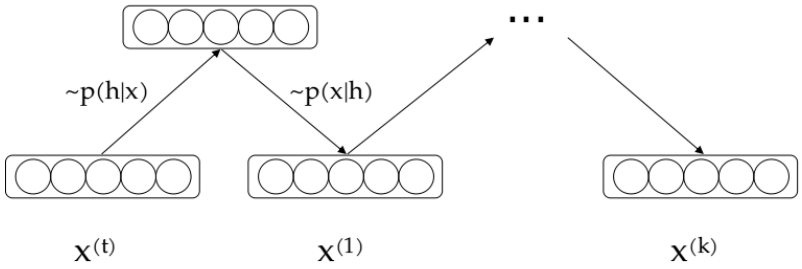


Figure 2.4: Illustration of Gibbs sampling

As illustrated in Figure 2.4, the conditional independence of intra-layer units in an RBM allows us to apply the Gibbs sampling iteratively for sampling visible and hidden units.

To perform the Gibbs sampling, the conditional distributions $P(\mathbf{h}|\mathbf{x})$ and $P(\mathbf{x}|\mathbf{h})$ have to be computed according to Equation 2.7.

$$\begin{aligned} P(\mathbf{h}|\mathbf{x}) &= \prod_i P(h_i|\mathbf{x}) \\ P(\mathbf{x}|\mathbf{h}) &= \prod_k P(x_k|\mathbf{h}) \end{aligned} \tag{2.7}$$

If $\mathbf{x}_k$ and $\mathbf{h}_j$ are binary units, the following applies

$$\begin{aligned} P(h_j = 1|\mathbf{x}) &= \frac{1}{1 + \exp(-(b_j + \mathbf{W}_{(j,:)}\mathbf{x}))} \\ &= \text{sigm}(b_j + \mathbf{W}_{(j,:)}\mathbf{x}) \\ P(x_k = 1|\mathbf{h}) &= \frac{1}{1 + \exp(-(c_k + \mathbf{h}^\top \mathbf{W}_{(:,k)}))} \\ &= \text{sigm}(c_k + \mathbf{h}^\top \mathbf{W}_{(:,k)}) \end{aligned} \tag{2.8}$$

After the negative sample $\mathbf{x}(k) = \tilde{\mathbf{x}}$ is estimated, the point estimate of the expectation in Equation 2.6 is computed as follows:

$$\begin{aligned} \mathbb{E}_{\mathbf{h}}\left[\frac{\partial E(\mathbf{x}^{(t)}, \mathbf{h})}{\partial \theta} \Big| \mathbf{x}^{(t)}\right] &\approx \frac{\partial E(\mathbf{x}^{(t)}, \tilde{\mathbf{h}})}{\partial \theta} \\ \mathbb{E}_{\mathbf{x},\mathbf{h}}\left[\frac{\partial E(\mathbf{x}, \mathbf{h})}{\partial \theta}\right] &\approx \frac{\partial E(\tilde{\mathbf{x}}, \tilde{\mathbf{h}})}{\partial \theta} \end{aligned} \tag{2.9}$$

Even with only one step of sampling, i.e., $k = 1$, this algorithm has been proven to be successful for unsupervised training [26]. Using Equation 2.9, the parameters θ of the RBMs are updated according to the following equation:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \left(\nabla_\theta (-\log P(\mathbf{x}^{(t)})) \right) \tag{2.10}$$

For each parameter $\mathbf{W}$, $\mathbf{b}$, and $\mathbf{c}$, the update equations are given below:

$$\begin{aligned} \mathbf{W}^{(t+1)} &= \mathbf{W}^{(t)} - \alpha \left(\mathbf{h}(\mathbf{x}^{(t)}) \mathbf{x}^{(t)^\top} - \mathbf{h}(\tilde{\mathbf{x}}) \tilde{\mathbf{x}}^\top \right) \right) \\ \mathbf{b}^{(t+1)} &= \mathbf{b}^{(t)} - \alpha \left(\mathbf{h}(\mathbf{x}^{(t)}) - \mathbf{h}(\tilde{\mathbf{x}}) \right) \\ \mathbf{c}^{(t+1)} &= \mathbf{c}^{(t)} - \alpha \left(\mathbf{x}^{(t)} - \tilde{\mathbf{x}} \right) \end{aligned} \tag{2.11}$$

where $\mathbf{h}(\mathbf{x}) \triangleq \text{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x})$.

Furthermore, having done with the unsupervised training, the parameters are used to initialize neural networks that will be trained using a supervised back-propagation algorithm. In this work, the neural networks are trained to learn several tasks at the same time, where each task consists of traffic flow prediction at each road. This type of learning is known as Multi-Task Learning (MTL) [72]. This learning method allows each task to share feature representation constructed by the stacked RBMs. Intuitively, MTL is suitable to be applied in a transportation network, where several freeways and stations are connected to each other. Thus, a huge amount of information is shared among freeways and stations. Figure 2.5 depicts a multi task regression stacked on top of the stacked RBMs.

2.4.2 Deep Neural Networks

A DNN is defined as an ANN with two or more hidden layers. It can be considered as stacked neural networks. The layers are composed of several nodes, which combine input with a set of weights. Furthermore, these combinations are summed before they are passed through activation functions. This sequence of processing is known as the forward pass. The subsequent step is to train the DNN with gradient-based learning methods and backpropagation [132]. In these methods, the weights are updated proportionally to the gradient of the loss function. There is a major problem when these methods are used for stacked neural networks; in some cases, the gradient will vanish, which is effectively preventing the weights from changing their values. This problem has been formally identified as the vanishing gradient problem [68]. One of the main causes is the use of traditional activation functions.

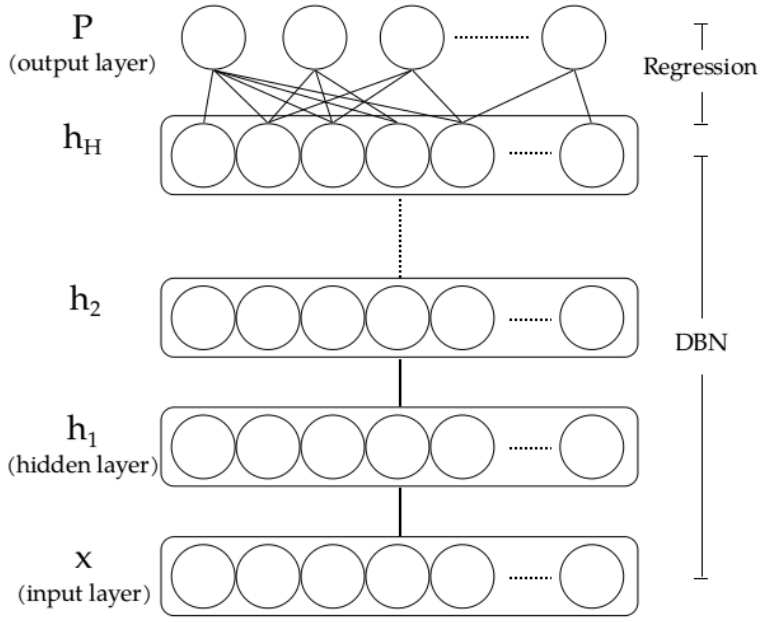


Figure 2.5: DBNs architecture for multi-task regression

Traditional activation functions such as the sigmoid function has derivative in the range of $(0, 0.25)$. The backpropagation algorithm computes the gradients using the chain rule, which directly translates to the multiplication of small numbers. In the deep network, the product of derivative will reduce the gradient very fast, which creates the vanishing gradient problem. One can avoid this problem by carefully selecting the activation functions; for example, Rectified Linear Units (ReLUs) [113]. These units output zero for negative inputs and linear positive values otherwise, which mathematically is defined as

$$\text{output} = \begin{cases} \text{input}, & \text{if input} \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.12}$$

According to [166], this simple activation function creates several advantages. First, it eliminates the necessity to pretrain the network [53]. Furthermore, it has been demonstrated that deep networks can be successfully trained using random initial weights and converge faster than the classical logistic function with similar configurations. Moreover, ReLUs are computationally cheaper because of their simple operations.

2.5 Dempster-Shafer Data Fusion

Data fusion deals with the synergistic combination of data and information from one or multiple sources to provide improved information with higher quality and reliability [76]. Applications or tasks that require parameters estimation from multiple sources can profit from data fusions techniques. While data fusion is used to handle raw data obtained from embedded sensors, information fusion handles data that are already pre-processed.

The state of the art provides several classification taxonomies of data fusion. One of the the most well-known classification is provided by Dasarathy [27]. Data fusion techniques are roughly classified into five categories.

- data in-data out (DAI-DAO): is the basic data fusion method. Data fusion is conducted immediately after the data are gathered from sensors. Both inputs and ouputs are raw data.
- data in-feature out (DAI-FEO): this category uses raw data from sources to extract features able to better describe an entity.
- feature in-feature out (FEI-FEO): is called also fusion or intermediate level fusion. Both inputs and outputs are features. The purpose of feature fusion is to improve existing one or obtain new features.
- feature in-decision out (FEI-DEO): the inputs of this category are features while the outputs are decisions. Hence, decisions are made based on features fed to the system.
- decision in-decision out (DEI-DEO): is called also decision fusion. Decisions are fused to obtain enhanced or new decisions.

The architecture proposed in this research will follow the last DEI-DEO model. The traffic flow provided by DBN using various sources will be fused to provide better decisions regarding traffic flow prediction accuracy. One of the methods for data fusion is DSET data fusion. This method was first introduced in [135]. The crucial feature of DSET is the combination of evidence obtained from several sources by modeling the conflict between evidence. Unlike Bayesian networks, DSET uses masses instead of probabilities. In the traffic prediction problem, evidence about traffic comes from two different types of data: hard data, e.g., traffic, weather, and soft data, e.g., Twitter data. The main decision about traffic is made from the combination of evidence collected from hard and soft data. Indeed, social media information presents several challenges such as reliability, uncertainty,

and unavailability in certain periods of time. Consequently, predicted traffic evidence from social media should support/update the predicted evidence about traffic from reliable data without compromising the integrity. Hence, fusing hard evidence with soft evidence is an issue for traffic prediction decision-making processes. For this purpose, the DSET evidence updating method for fusing hard evidence and soft evidence is studied.

In DSET, the total set of mutually exclusive and exhaustive propositions of interest that a decision authority may discern, is called the Frame of Discernment (FoD). It is the set of possible conclusions to be drawn. Let $\theta = \{\theta_1 .... \theta_n\}$ be an FoD. A singleton proposition θ_i denotes the lowest level of discernible information. The power set of θ is the set of all possible subsets of θ. Given $|\theta| = n$, the power set denoted by 2^θ contains 2^n elements that are composed of all the subsets of θ. In the following, $\bar{A}$ denotes all singletons in θ that are not included in A. The Basic Belief Assignment (BBA) or mass assignment provides the support for proposition A.

Definition 1 *The mapping m:* $2^\theta \mapsto [0, 1]$ *is a BBA for the frame* θ *if*

$$m(\emptyset) = 0 \text{ and } \sum_{A \subseteq \theta} = 1 \tag{2.13}$$

The set of propositions in a frame θ that possess nonzero BBAs or masses are called the focal elements of θ, and is denoted by $\mathcal{F} = \{A \subseteq \theta : m(A) > 0\}$. The triple variables $\{\theta,\mathcal{F}, m\}$ is called the Body of Evidence (BoE). For any event A, all testimonies in its favor can be collected and the *belief function* which corresponds to its can be determined. By taking into account all the focal elements related to A, its *plausibility function* can be determined. Consider the following definition.

Definition 2 *Given a BoE* $\{\theta, \mathcal{F}, m\}$ *and* $A \subseteq \theta$*, Belief and Plausibility functions can be defined as:*

$$\begin{aligned} Bel(A) &= \sum_{X \subseteq A} m(X) \\ Pl(A) &= \sum_{X \cap A \neq \emptyset} m(X) \end{aligned} \tag{2.14}$$

While $m(A)$ measures the support assigned to proposition A only, the belief assigned to A measures the supports for all proper subsets of A. The plausibility function can also be derived from the belief function as follows

$$Pl(A) = 1 - Bel(\bar{A}) \tag{2.15}$$

In DSET, the uncertainty interval associated with A is bounded by belief and plausibility, i.e., $Un(A) = [Bel(A) \;\; Pl(A)]$. This interval can be regarded as framing the poorly known probability $P(A)$ as the following

$$Bel(A) \leq P(A) \leq Pl(A), \quad \forall A \in \theta \tag{2.16}$$

2.5.1 Dempster's Rule of Combination

In DSET, the Dempster's Rule of Combination (DRC) is used to find a new BBA that aggregates the evidence generated by several BoEs spanning the same FoD. It can be seen as several experts indicating their opinion via the attribution of masses over θ. The DRC highlights combined masses of belief in the following way:

Definition 3 *Given* $BoE_i = \{\theta_i, \mathcal{F}_i, m_i\}$, *where* $i = 1, 2$ *and* $\theta = \theta_1 = \theta_2$, *the DRC generates the following*

$$BBA_{12} \equiv BoE_1 \bigoplus BoE_2 = \{\theta_{12}, \mathcal{F}_{12}, m_{12}(\cdot)\} \tag{2.17}$$

$$m_{12}(A) = \frac{\sum_{B,C:B\cap C=A} m_1(B) \cdot m_2(C)}{(1 - \sum_{B,C:B\cap C=\emptyset} m_1(B)m_2(C))}, \;\; \forall B, C \subseteq \theta \tag{2.18}$$

It is worth noting that $\sum_{B,C:B\cap C=\emptyset} m_1(B)m_2(C) \in [0, 1]$. The denominator is a coefficient of normalization. In particular, if it is null, it means that there is a total conflict between the sources, and aggregation is then impossible. The use of this rule is thus valid only when the sources are sufficiently in agreement. Moreover, this rule is suitable when two distinct beliefs are always certain and treated equivalently. When these are not the case, a method to incorporate this notion has to be investigated; one way is to use Dempster's rule of conditioning [135].

2.5.2 Conditioning rules for updating belief

The first and well-known conditioning rule in the theory is Dempster's rule of conditioning [135], which is derived as a special case of Dempster's rule of combination. When the BBAs in the DRCs is given as $m_1(\cdot) = m(\cdot)$ and $m_2(C) = 1$, the DRCs is transformed into the following

$$m(A|C) = \frac{\sum_{A=B\cap C} m(B)}{1 - \sum_{B\cap C=\emptyset} m(B)} \tag{2.19}$$

In [160], the authors generalize the rule of conditioning by differentiating two cases: conditioning rule with unreliable condition and conditioning rule with uncertain condition. The basic conditioning rule assumes the following:

> *The given condition is exactly true information given temporally after the prior belief. It puts a strict restriction on focal elements of the posterior belief* [160].

First, the an index λ ($0 \leq \lambda \leq 1$) is introduced to represent reliability of the condition. When $\lambda < 1$, the condition is not completely true. The mass function $m(B|A)$ is given by:

$$m(B|A) = \begin{cases} \sum\limits_{A=X\cap B} \lambda m(X) + (1-\lambda)m(A) & \text{if } B \supset A \neq \emptyset \\ \sum\limits_{A=B\subseteq X} \lambda m(X) + (1-\lambda)m(B) + \sum\limits_{X\cap B=\emptyset} \lambda m(X) & \text{if } A = B \\ (1-\lambda)m(A) & \text{if } A \nsubseteq B, A \neq B, A \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{2.20}$$

where $B \neq \emptyset$. It can be proved that $m(A|B)$ is a BBAs [160].

2.5.3 DSET Data Fusion in the Literature

In smart cities environment, exploiting diversities in data, i.e., data fusion, to improve decision-making processes is important. A critical issue for this type of decision-making is the uncertainty of the data, which may propagate to or affect the decision made by the fusion methods. There are three main theories to handle uncertainty in data: probability, possibility, and belief theories. In this work, a belief theory, namely DSET [135], is used to give a formal representation of the inaccurate and uncertain aspect of information provided by the proposed prediction model. The following paragraphs present a literature review in the recent application of DSET for data fusion and the variation or extension of the theory.

There are several applications of DSET application for data fusion in the literature. In [48], DSET is used to yield a global judgment of several different image forensic tools. In a different area, vibration and acoustic signals for fault diagnosis and classification of planetary gears are fused with DSET [78]. An industrial engineering application of a multiple-criteria decision-making based on fuzzy evidential theory investigated to select

plant location is presented in [38]. In [100], DSET is proposed as the fusion framework to combine ultrasonic micro-Doppler and passive infra-red sensors to detect personnel intrusion. This framework assists surveillance systems to classify between a human and a horse.

In the Intelligent Transportation System (ITS) area, DSET has been used to enhance travel time estimations by fusing inductive loop and toll collection data [43]. In another application, the combination of Hierarchy Process and DSET are used to create a prioritized performance hierarchy, i.e., the ITS sustainability index [83]. In [129] and [37], multiple feature analysis and Dempster-Shafer fusion theory is used to detect vehicle and pedestrian respectively. Finally, DSET-based traffic speed estimation using heterogeneous sources for first response deployment application is proposed [33]. To the best of our knowledge, although there are several applications of DSET in ITSs, there is no implementation of DSET to fuse information coming from transportation big data for traffic flow prediction.

2.6 Summary and Highlights

Planning proactive operations to alleviate road congestion and guiding drivers to better plan their trip are important requirements for traffic management systems. Therefore, a big data-based traffic prediction model that utilizes the various information available in IoC, to enhance the quality of the decision-making is among the main topics of contemporary research for achieving this goal.

This chapter presents an overview on the concept of smart city and traffic flow prediction. Furthermore, it outlines the need for having a big data-driven computer-based traffic prediction model. Furthermore, big data-based traffic flow prediction is introduced as a promising methodology that can be used in an environment characterized by diverse types of data and information. Moreover, some comprehensive traffic flow prediction methods that appeared in the literature are briefly introduced, and features are specified. However, most of these methods face some challenges in transportation systems with big data framework, among which the assumption made in deriving the physical equations.

The data-driven modeling approaches are mainly addressed in the literature from the type of model point of view, i.e., parametric, non-parametric, and hybrid approaches. Most of the approaches either require a prior knowledge of specific domains for feature extraction and selection or are shallow in architecture. Several state-of-the-art results have been presented to showcase how far has the traffic prediction technology progressed. However, the main deficiency of many of these methods is their inability in incorporating

various sources to increase the accuracy of the prediction. Previous work on non-stationary traffic flow prediction and multi-step traffic flow prediction has been discussed. It can be concluded that both of these prediction problems are still a major challenge.

Deep learning methods are selected to be the main predictors in this thesis. Therefore we provide a concise overview on deep learning, more specifically DBN and DNN. The technical details on DBN and DNN are discussed, including the modern approach to make the deep architecture works efficiently. Furthermore, the advantages on using DBN and DNN are highlighted.

A part of this chapter has been dedicated to the investigation of the common uncertainty handling techniques in data fusion. In particular, DSET to deal with the inherent uncertainty in the information that ranges from the input data to the output model. In other words, it is shown that dealing with uncertainty is imperative as it can greatly affect the quality of the prediction greatly. Therefore, among the different approaches confronting uncertainty, DSET had been shown to be the reliable method in handling imprecise probability.

Various studies on traffic flow prediction affirm that there is a missing link in the complete picture of big-data-driven smart-mobility enabled traffic prediction. Accordingly, the core of the model presented in this thesis is developed based on deep learning methods.

Chapter 3

Large-Scale Traffic Flow Prediction[1]

3.1 Introduction

Traffic flow prediction is a challenging problem. During trips, drivers face several inclement weather conditions, e.g., rain, storm and snow, and unforeseen events, e.g., accidents, concerts, and road closures. These factors impact significantly the travel time of each driver, which consequently affect the overall traffic flow of transportation networks. Furthermore, connected cars evolve in a data-rich environment, where they consistently generate and receive a variety of data. Citizens also produce huge streams of data using social media. While driving, citizens announce their travel plans and share information regarding current traffic or road conditions using Twitter, Facebook, or Waze. This information can be used to enrich our experiences on how cities function and as a fuel that drives predictive related traffic applications such as traffic flow prediction and informed decision-making.

While the prediction of traffic using streams of data has received a lot of attention, the technical capacity of social media, as event-based data, to provide insights on traffic flow is only just now catching up. The present work proposes a Deep Belief Network (DBN) based approach to predict traffic flow using streams of data, i.e., historical traffic flow

[1]This chapter contains two published articles. The first article is published in IEEE Transaction on Vehicular Technology on June 28, 2016, available online: `https://doi.org/10.1109/TVT.2016.2585575`. Arief Koesdwiady, Ridha Shoua, and Fakhri Karray, "Improving Traffic Flow Prediction With Weather Information in Connected Cars: A Deep Learning Approach" [82]. The second article is published in the proceeding of International Joint Conference on Neural Networks (IJDNN) 2016, available online: `https://doi.org/10.1109/IJCNN.2016.7727607`. Ridha Soua, Arief Koesdwiady, and Fakhri Karray, "Big-Data-Generated Traffic Flow Prediction Using Deep Learning and Dempster-Shafer Theory" [137].

and weather data, and event-based data, i.e., tweets. Furthermore, a novel architecture to model and analyze the traffic prediction in a smart city is presented. First, a generic scheme to fuse streams of data and event-based data is illustrated. In this part, a novel six-layered traffic flow prediction for data streams and event-based data is proposed. Subsequently, an extension of Dempster Shafer Evidence Theory (DSET), namely Dempsters conditional rules for updating belief, is used to fuse traffic prediction beliefs coming from streams of data and event-based data modules.

The results of applying the proposed architecture to a big-data-based traffic prediction problem are also presented. First, the experimental settings that include the data-sets used in the experiments are provided. Then, the streams of data processing and prediction are introduced. Normalization of the data, as well as correlation study, are conducted. The event-based data processing and prediction, which include traffic-related city events extraction, are then assessed. Furthermore, the results regarding the combination of streams of data and event-based data using Dempsters theory are discussed. Several performance measures to check on the accuracy of the proposed model are introduced; the results are analyzed to discover which components of the model need to be optimized or enhanced.

3.2 Proposed Architecture

While the increased availability of a multitude of streaming/stored sensor and data feeds has made the tasks of coordinated, persistent, and pervasive surveillance and decision making easier, it has also been the major contributing factor for the immense burden that traffic authorities have to deal with. To ease this burden, increased automation of the decision-making process is warranted. This, in turn, calls for effective, computationally viable, and analytically tractable models for data representation and schemes for evidence fusion. There are two main schemes for big-data-based traffic prediction architecture proposed in this research. The first one is based on the type of data generation. Basically, various sources of data are categorized into two types: streams of data and event-based data. Data that are categorized as streams of data are data that are continuously generated by sensors; for example, traffic flow data and weather data. Event-based data refers to data that are generated based on events; soft data are generally considered as event-based data. The enormous challenge one faces is mainly due to the critical issue in fusing streams of data and event-based data. Another challenge is related to the types of data: How should more "qualitative" information provided by soft sensors be fused with the more "quantitative" information generated from the hard sensors? This is an issue that has recently attracted considerable attention from the data and evidence fusion community [77].

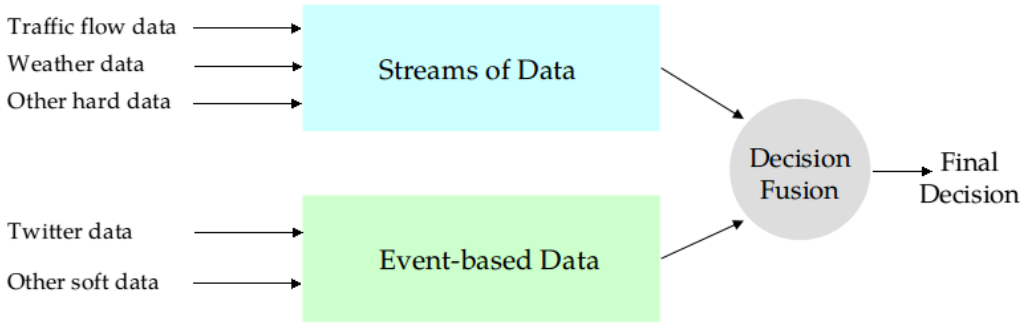


Figure 3.1: Streams of data and event-based data fusion.

In this research, a decision level fusion architecture based on Dempster-Shafer Evidence Theory (DSET) is proposed. The streams of data and event-based data are processed separately. Each type of data is used to predict traffic flow, and the decisions generated by each prediction model are fused using DSET. This scheme is depicted in Figure 3.1. The final decision generated by this system is then used by travelers or authorities to improve traffic planning and management.

The second scheme describes the details of the processing of each type of data. This scheme illustrates the six-layered traffic flow prediction model in smart city, and describes the essential key technologies used which include transportation resource layer, pre-processing layer, feature selection layer, prediction layer, decision fusion layer, traffic clustering layer, and mass assignment layer. The technology implementation of this scheme has to be performed in sequence, as shown in Figure 3.2. The key functions of each layer are described next:

- Resource layer–This layer is responsible for establishing connection to variety of data sources or suppliers such as traffic data stations, weather stations, transportation authorities, and other relevant sources. This layer tackles several key features of big data: volume, exhaustive, and relational.

- Preprocessing and feature selection layer–This layer first identifies good features for the prediction and normalizes the data if needed. This layer captures the *value* notion of big data.

- Prediction layer–This layer is responsible for exploiting the selected and normalized features in the traffic flow prediction. If the features are selected and arranged

properly, it can capture the resolution, velocity, volume, and exhaustive aspects of big data.

- Decision fusion layer–This layer enables the variety and relational aspects of big data to be tackled by fusing the decisions coming from the predictors, which use different data sets and type of data for the traffic flow prediction.

- Traffic clustering layers–This layer is responsible of transforming the multi-valued traffic flow volume into levels of traffic flow volume. This is done because a finite number of evidence is needed when fusing the hard and soft data using DSET. This kind of information is more useful to the travelers or authorities in the decision making.

- Mass assignment layer–In this layer, the evidence in the form of Basic Belief Assignment (BBA) is constructed. Using the BBA, the Dempster's Rule of Combination (DRC) or Dempster's rule of conditioning can be applied.

The main reasons for proposing such schemes are the modularity and the real-time implementation aspects. When another source of information is introduced in the system, it can be easily integrated into the architecture. These layers also allow for real-time implementation of the proposed model. The modularity and real-time aspects are in accordance with the fast and flexible nature of big data. In addition, by having the modularity, each part of the model can be tuned or optimized separately to achieve improved traffic flow prediction. The following sections elaborate the detailed implementations of the two schemes.

3.3 Streams of Data

This section presents the development of the traffic prediction modeling using streams of data. After the sources of information are established in the resource layer, i.e., traffic flow and weather data, the next step is to pre-process the data and select pertinent features for the prediction. In this layer, the traffic flow and weather data are normalized to [0,1]. One component of the collected weather data is the weather condition, which is considered as soft data generated by operators. Although the weather condition is considered as soft data, it will be treated as streams of data since it is available continuously at every

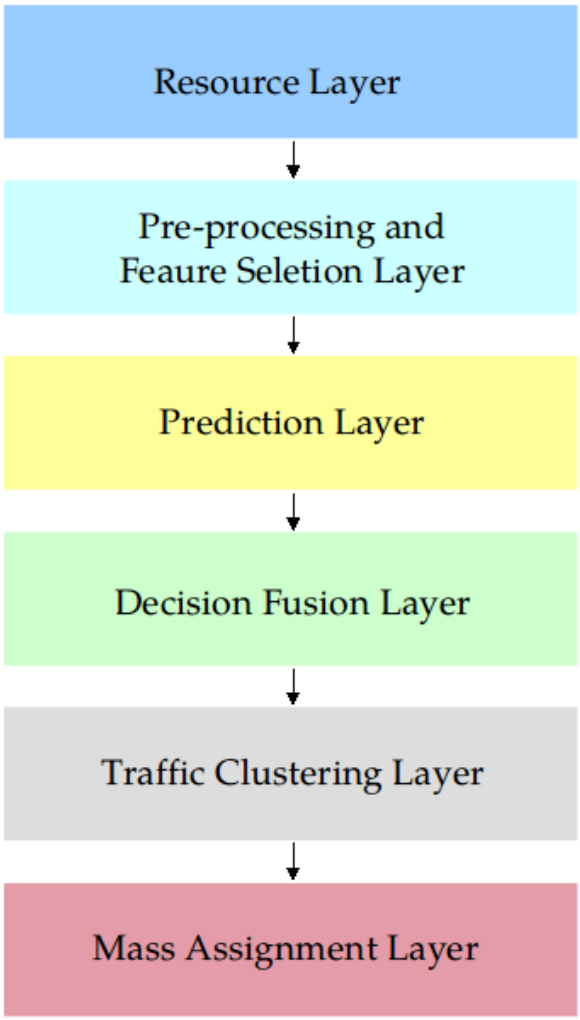


Figure 3.2: Layers for data processing.

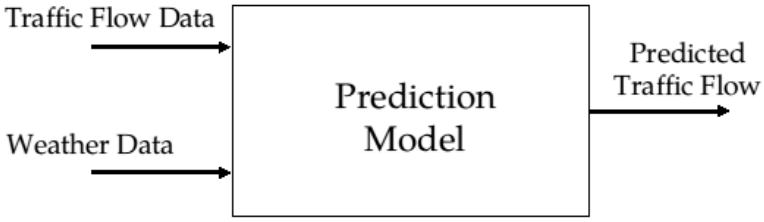


Figure 3.3: Feature-Level data fusion for traffic flow prediction using weather data.

sampling time. Weather condition consists of 24 unique values, which are represented by:

$$\begin{aligned} \text{weath\_cond} = \{&clear, cloudy, foggy, rain, \\ &snow, overcast, smoke, ...\} \end{aligned} \tag{3.1}$$

In order to incorporate the weather conditions in the prediction, the soft-data representation is re-defined as an integer number in the range of $[1, 10]$, for instance $clear = 1$, $cloudy = 2$.

In the next layer, feature selection is done by auto- and cross-correlation methods. Correlation is crucial in this study as huge traffic data are collected every fifteen minutes for 4 months from 47 freeways. In addition, soft and hard weather data are collected from 16 stations scattered in the Bay Area. In the context of smart mobility, computing correlation involving a huge number of paired time-series (weather and traffic), ensures pertinent feature selection and best prediction accuracy [41]. The feature selection methods will select relevant information and discard redundant information. In this work, the auto-correlation coefficients of traffic flow in freeways in San Francisco, Bay Area is calculated. Subsequently, the cross-correlation between traffic flow and various weather parameters time-series (humidity, wind speed, temperature, etc) in the same area are analyzed. After the pertinent traffic flow and weather features are selected, the features are then fed to the predictors in the prediction layer.

The predictor used in this research is DBNs [67]. DBNs can be used as a predictor or a data fusion method; the use of DBNs as a predictor and a data fusion method is investigated in this work. Indeed, the traffic flow prediction problem is a regression problem. In the multi-task part of the DBNs, namely the last layer of the DBNs, a linear activation function is used instead of some nonlinear activation functions. First, the traffic flow and weather data are used together as features for the DBNs, as depicted in Figure 3.3.

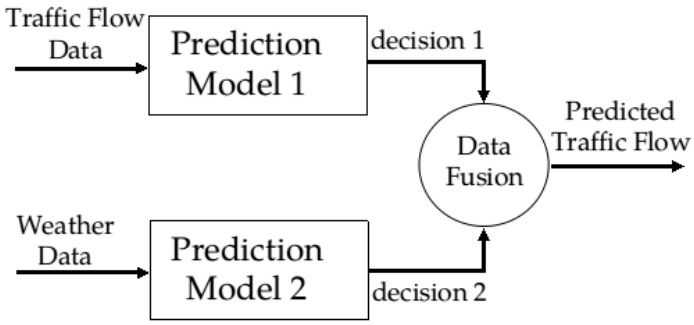


Figure 3.4: Decision-Level data fusion for traffic flow prediction using weather data.

In this model, the input of the prediction model is the historical traffic flow and weather data, and the output is the traffic flow data. The historical traffic flow data contain current traffic flow data and previous traffic flow data up to k_1-step in the past. The traffic flow data are collected from 47 roads and used simultaneously in the prediction. After the initial weights of the DBNs are learned by the unsupervised stacked Restricted Bolztmann Machines (RBMs), a supervised back-propagation algorithm is implemented to tune the weights of the DBNs.

A decision-level data fusion for traffic flow prediction using weather data is also introduced in this research. In this architecture, the traffic flow and weather data are predicted separately using DBNs, as illustrated in Figure 3.4. In this type of system, the k_1-step in the past traffic flow data are trained using k_2-step in the future traffic flow data as the output. Similarly, the current weather data are trained using k_2-step in the future traffic flow data as the output. Both of the data are trained using DBNs similar to the feature-level data fusion method. The result of each the predictors is then merged using data fusion techniques. The techniques fuse a set of decisions produced by each prediction model to obtain enhanced new traffic flow prediction. This type of fusion scheme follows the DEI-DEO scheme discussed in Chapter II.

In the data fusion part of the architecture, several methods of data fusion are tested: weighted average, Artificial Neural Network (ANN), and DBN. In the former technique, the output of each prediction model is weighted to produce better traffic flow predictions. These weights are calculated using least-square estimate method according to the following equations:

$$Y = \Phi^\top W$$
$$\hat{W} = (\Phi^\top \Phi)^{-1} \Phi^\top Y \tag{3.2}$$

where

$$Y \triangleq \text{output of the data fusion}$$
$$X \triangleq \text{input of the data fusion}$$
$$W \triangleq \text{the weights of the data fusion inputs}$$

In the ANN, the decisions coming from both predictors are trained using back-propagation algorithm with the actual traffic flow at k_2-step in the future as the output. Similar to the DBN, instead of using ANN, the fusion method is replaced by a ANN.

After the predicted traffic flow data are obtained, the next processing layer is the traffic clustering layer. The aim of using this layer is to cluster the traffic flow based on the level of intensity. In this research, the traffic flow data are clustered into four clusters: low, medium, high, and very high traffic. The method used to cluster the data is the well-known K-Means clustering. The K-Means clustering algorithm aims to partition the data into K clusters in which each observation belongs to the cluster with the nearest mean or center. In the training, the algorithm assigns each data point to a cluster and adjusted the centers of the clusters until they converge. K-means is considered as a hard clustering technique since the result of the clustering belongs to only one cluster. These clusters then are used to cluster new data and will be used in the mass assignment layer to compute the masses of each sample.

In the mass assignment layer, the distance of each data point is calculated. The following assumption is used to compute the masses:

Assumption 1 *The prediction results of the predictors are assumed to be accurate enough, such that the misclustering can occur only in the neighboring clusters.*

Suppose a test sample belongs to cluster *medium traffic*. When its predicted sample is available and clustered, it can either be clustered as *low traffic* or *high traffic*. The misclustering is occurred due to the prediction error that propagates to the clustering process. Following the assumption, after the distances of a sample to the centers are

calculated, then the two lowest distances are selected. The selected two distances then are converted into some confidence values as the following:

$$C(d) = \exp(-\alpha d) \tag{3.3}$$

where $C(d)$ is the confidence value, $d(0 \leq d \leq 1)$ is the distance of a sample to a center, and $\alpha(0 \leq \alpha \leq 1)$ is used to control the weight of the exponential function. This function translates the distance to confidence value non-linearly. If the distance of a sample to a center is small, it can be said with high confidence that the sample belongs to the center. The α can be used to control the significance of the distance. Once the confidence values of the two smallest distances are computed, the difference of the confidence values is then calculated. There are two possible cases for the difference values: the difference is within a specified small tolerance value, i.e., there is a confusion in the cluster assignment, or the difference is quite large, i.e, the sample belongs to the cluster with the highest confidence. In the former case, the confusion can be modeled by considering the boundary of the two neighboring clusters as an evidence. These cases determine the number of focal elements in the power set. Based on the number of clusters, the number of elements in the power set is $2^4 = 16$. Consider the following example: the confidence of cluster *low traffic* $C(d_L) = 0.41$, and the confidence of cluster *medium traffic* $C(d_M) = 0.4$. If the difference tolerance is 0.2, then the following is computed:

$$C(d_{LM}) = C(d_L) + C(d_M) - |C(d_L) - C(d_M)| = 0.8 \tag{3.4}$$

The summation term denotes the confidence value assigned to the boundary of the cluster, and the absolute difference term denotes the uncertainty in the boundary. The three confidence values then are marginalized to 1; hence the final confidence values are:

$$C(d_L) = 0.2547, \;\; C(d_M) = 0.2484, \;\; C(d_{LM}) = 0.4969 \tag{3.5}$$

This equation illustrates the adjusted confidence values of the *low traffic* and *medium traffic* clusters, and assigns a confidence value the boundary. The other elements in the power set are assigned zero masses. Indeed, when a sample is in fact in the boundary, it has the highest confidence compared to the other clusters. In the second case, only the two highest confidence values are considered. These values are then marginalized to 1, and other elements in the power set are assigned zero masses. This method of mass assignment is illustrated in Algorithm 1.

Algorithm 1 Mass assignment algorithm

Input: two smallest distances
Output: set of masses
Initialization: difference tolerance, α
for all data points **do**
  Compute the the distance difference $|d_1 - d_2|$
  for $i = 1 : 2$ **do**
    $C(d_i) = \exp(-\alpha d_i)$
  end for
  if $|d_1 - d_2| \leq$ difference tolerance **then**
    $C(d_{12}) = C(d_1) + C(d_2) - |C(d_1) - C(d_2)|$
    marginalize $C(d_{12}), C(d_1), C(d_2)$ into masses m_1, m_2, m_{12}
    $m_{\text{others}} = 0$
  else
    marginalize $C(d_1), C(d_2)$ into masses m_1, m_2
    $m_{\text{others}} = 0$
  end if
  return m_{all}
end for

3.4 Event-Based Data

In this section, a detailed implementation of the event-based data processing is presented. The steps of the implementation still follow the six-layered procedure. The main difference between the event-based data processing and the streams of data processing lies in layer one and two, whereas the rest is similar. In the first layer, Twitter data are established as the source of information. However, based on our experiments, Twitter data can not stand by themselves without the information from traffic flow data. Therefore, the traffic flow data are included in the first layer.

In the pre-processing and feature selection layer, the traffic flow data are normalized, and the pertinent features are selected using auto-correlation. Subsequently, the Twitter data are pre-processed from soft data to hard data by means of annotation and extraction. Social media data processing is an emerging and challenging research area, and it requires intensive research efforts. Furthermore, since the scope of this research does not include the social media data processing, tools of annotation and extraction provided in [11] are adapted.

In [11], two-steps traffic city events extraction are presented. In the first step, Twitter data are annotated by employing a sequence labeling task using Conditional Random Field (CRF). The objective of this task is to spot event and location terms in tweets. The main challenge in identifying locations in Twitter data is the presence of non-standard abbreviations, spellings, and capitalization convention. To overcome this challenge, the sequence model is trained using the knowledge of locations from Open Street Maps for a specific city. Another challenge is identifying event terms in Twitter data due to the open domain nature of city related events. Therefore, background knowledge consisting of domain vocabulary is obtained from 511.org, which provides a hierarchical classification of traffic related events. Once the CRF model is trained, the next task is to extract traffic related city events from the Twitter data.

There are two algorithms used to extract the traffic events. The first algorithm obtains n, e, t, d, l, which denote the location terms, event terms, time of day, day of week, and geohash location, of the Twitter data using the trained CRF model. The algorithm for this task is presented in Algorithm 2.

The second algorithm derives traffic-related city events from the feature vectors, i.e., n, e, t, d, l, and returns the extracted feature vectors $(\hat{e}_{i,type}, \hat{e}_{i,loc}, \hat{e}_{i,st}, \hat{e}_{i,et}, \hat{e}_{i,impact})$. $\hat{e}_{i,type}$ ranges over all the traffic events extracted from Twitter. $\hat{e}_{i,loc}$ refers to the location of the event (lat-long), $\hat{e}_{i,st}$ and $\hat{e}_{i,et}$ refers to the start time, and the end time of the event, and $\hat{e}_{i,impact}$ refers to a number quantifying the severity of the event.The details of the algorithm

Algorithm 2 Populating metadata for each tweet

Input: tweets, time stamp, latitude, longitude, CRF model
Output: n, e, t, d, l
n:= spotEntities(tweets, time stamp, latitude, longitude, CRF model)
e:= spotEventTerms(tweets, time stamp, latitude, longitude, CRF model)
t:= getTimeOfDay(tweets, time stamp, latitude, longitude)
d:= getDayOfWeek(tweets, time stamp, latitude, longitude)
l:= getGridNumberFromGeohash(tweets, time stamp, latitude, longitude)
return n, e, t, d, l

are depicted by Algorithm 3.

The extracted feature vectors produced by this algorithm then are used for the prediction. The start and end time features are utilized to align the data with other types of data. However, one of the main characteristics of event-based data is that they are not continuously available. In the first stage of this research, the unavailable data at certain time steps are estimated using Principal Component Analysis with Alternating Least Square (PCA-ALS) algorithm. Principal Component Analysis (PCA) is a classical data analysis technique that finds linear transformations of data that retain the maximal amount of variance. PCA with missing values is studied in [57]; the authors propose to use the faster alternating W-X algorithm, which is known as PCA-ALS. The PCA-ALS algorithm is developed based on missing data assumption rather than unavailable data assumption. However, the implementation proposed in this research applies the PCA-ALS to unavailable data. The incompatibility of the implementation, as well as the unreliability of the Twitter data, will be used as unreliability indicators in the streams of data and event-based data fusion.

In the prediction layer, the traffic flow and extracted Twitter data are concatenated and used as features for the DBN. The output of the DBN is the traffic flow at k_2-step in the future. The traffic clustering and mass assignment layers follow the same methods in the streams of data processing.

3.5 Fusion of Streams of and Event-Based Data

This section presents the proposed method used to fuse decisions coming from both streams of data and event-based data processing units. Dempster's conditioning rules for updating

Algorithm 3 Traffic-related city events extraction algorithm

Input: $(n, e, t, d, l)_N, \Delta t, t_s, t_e$, where N refers to the number of input points, Δt represents the time step, t_s represents starting time, and t_e represents ending time
Output: $(\hat{e}_{i,type}, \hat{e}_{i,loc}, \hat{e}_{i,st}, \hat{e}_{i,et}, \hat{e}_{i,impact})_E$, where E represents the number of events
while $\exists(n, e, t, d, l)$ **do**
  for $i = 1 : N$ **do**
    $v_i = (n, e, t, d, l)$
    $type =$ 511.org hierarchy term associated with e
    Assign $type$ to v_i
  end for
  for $i = 1 : N$ **do**
    Collect all the feature vectors v_i with the same type into an event type bucket $E[type_k]$ where k is the number of event types
  end for
  for $i := 1 : k$ **do**
    Find location with highest number of occurrence in $E[type_i]$ represented by $lmax$; remove all the members of the set $E[type_i]$ whose location is not $lmax$;
  end for
  for $i := 1 : k$ **do**
    $\hat{e}_{i,type} = type$
    $\hat{e}_{i,loc} = lmax$ associate with $E[type_i]$
    $\hat{e}_{i,impact}$ = number of items in the set $E[type_i]$
    $\hat{e}_{i,st}$ = smallest time stamp in the cluster $E[type_i]$
    $\hat{e}_{i,et}$ = smallest time stamp in the cluster $E[type_i]$
  end for
end while

belief is employed, and the adaptive version based on the reliability of the data is proposed. Using equation 2.20, the following algorithm is proposed.

This algorithm computes the mass of an event conditioned by another event. The reliability factor $\lambda(t)$ determines the contribution of new information to the existing evidence. In this work, tweets are considered as the new information that conditions the fundamental evidence provided by the streams of data processing. The fact that the reliability of the information depends on events occurred implies that $\lambda(t)$ changes over time. The overall implementation of the proposed architecture is depicted in Figure 3.5.

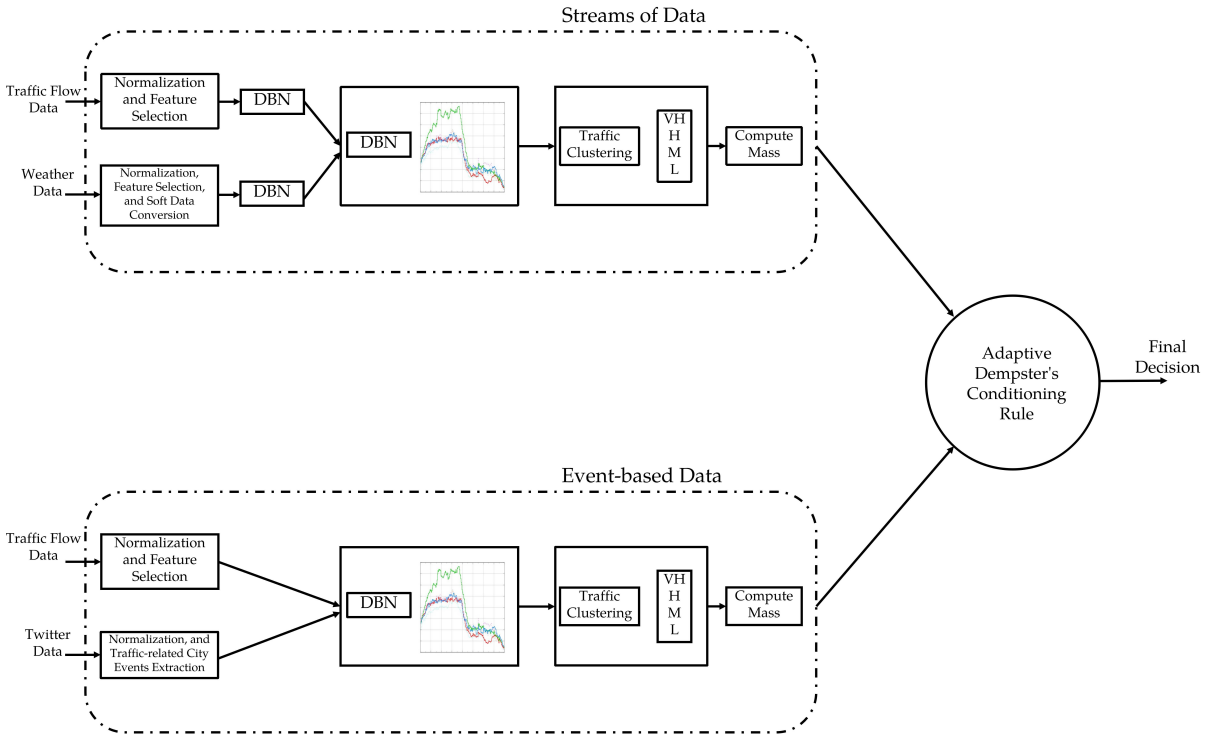


Figure 3.5: Overall proposed architecture.

Algorithm 4 Adaptive Dempster's conditioning rules

Input: decisions coming from streams of data and event-based data processing units
Output: $m(B|A)$
for all roads **do**
  for all each data point at t **do**
    Compute the reliability parameter $\lambda(t)$ based on the estimated features.
    if $B \supset A \neq \emptyset$ **then**
      $m(B|A) = \sum_{A=X\cap B} \lambda(t)m(X) + (1 - lambda(t))m(A)$
    else if $A = B$ **then**
      $m(B|A) = \sum_{A=B\subseteq X} \lambda(t)m(X) + (1 - \lambda(t))m(B) + \sum_{X\cap B=\emptyset} \lambda(t)m(X)$
    else if $A \nsubseteq B, A \neq \bar{B}, A \neq \emptyset$ **then**
      $(m(B|A) = 1 - \lambda(t))m(A)$
    else
      $m(B|A) = 0$
    end if
  end for
end for

3.6 Experimental Settings

In this section, the data-sets and settings of the experiments are described. These data-sets are used to perform the correlation analysis as well as training and testing of the DBNs. The settings allow us to show the merit of the proposed methods, decision-level data fusion architecture, and algorithms for big-data-based traffic flow prediction. In addition, the performance indices used in this experiment are presented at the end of this section.

3.6.1 Datasets

In this study, the collected traffic flow, weather, and Twitter data are specific to an area in California, i.e., Bay Area, District 4, San Francisco, as shown in Figure 3.6. Traffic flow and weather datasets in this area are obtained from the Caltrans Performance Measurement System (PeMS) [1] and MesoWest project respectively [4], where as the Twitter data are collected from CityPulse data-set collection[2].

The traffic flow is measured using inductive-loop sensors deployed throughout the free-ways. These sensors are connected to the road-side controllers from which the traffic flow

[2]http://harp.cs.wright.edu/cityevents/

Figure 3.6: District 4, Bay Area, San Francisco (Image courtesy of dot.gov.ca)

Table 3.1: Road numbers and directions

Road Number	Direction	Road Number	Direction
1	S	101	N,S
4	E,W	152	E,W
17	N,S	237	E,W
24	E,W	238	N,S
25	N,S	242	N,S
29	N,S	280	N,S
37	E,W	580	E,W
80	E,W	680	N,S
84	E,W	780	E,W
85	N,S	880	N,S
87	N,S	948	N,S
92	E,W	980	E,W

data will be sent to the District Transportation Management Center via modem every 30 seconds. These data then are aggregated into 5-minute duration by PeMS. Furthermore, based on the recommendation of Highway Capacity Manual 2010 [102], the traffic data are aggregated further into 15-minute duration. In the collected dataset, there are 47 roads, where the traffic flow of each road is calculated based on the average of all the loop detectors in that particular road. Table 3.1 presents the numbers and directions of the freeways in our study.

Figure 3.7 plots the 2-weeks traffic flow patterns in selected freeways with low, medium, and high traffic flow. These categories are manually clustered based on the mean of the traffic flow in the 47 freeways. As can be seen in the figure, traffic flow witnesses peak points during rush hours and reaches its lowest point during the night.

The weather dataset is obtained from 16 National Weather Service network stations scattered throughout the district, as summarized in Table 3.2. Each station provides data about the temperature, humidity, visibility, wind speed and gust, dew point, cloud layer height and general weather condition. The weather stations are located in close proximity of the chosen freeways. The weather data are generated every one hour and then re-sampled into 15-minute duration for the purpose of alignment with the traffic flow data. A portion of temperature data generated by KSQL weather station is depicted in Figure(3.8)

The inputs for the prediction models are the traffic flow of all roads at the previous

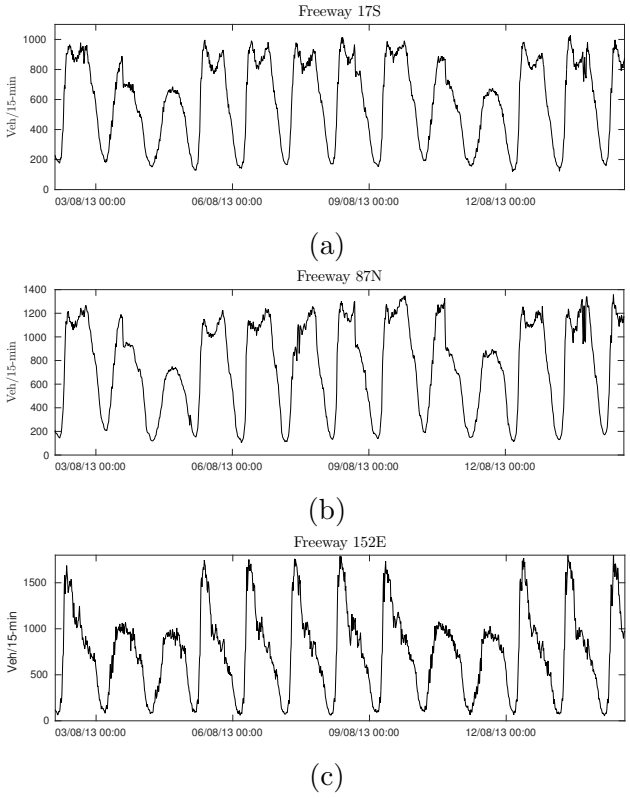


Figure 3.7: A portion of traffic flow data with different traffic volume. Freeway with low (a), medium (b), and high (c) traffic flow.

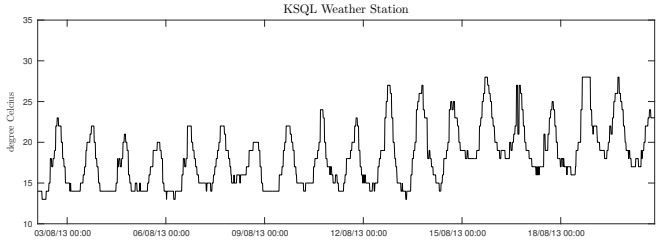


Figure 3.8: A portion of temperature data generated by KSQL weather station.

Table 3.2: National weather stations in district 4, San Francisco

Station Name	Station Name
KAPC	KHWD
KMRY	KNUQ
KOAK	KSFO
KSJC	KSNS
KSTS	KWVI
KDVO	KE16
KHAF	KRHV
KSQL	KPAO

k-step intervals. and weather data from 16 stations at the previous 1-step intervals. For the weather data, we select only 1-step interval since the traffic flow prediction will be mainly based on the past traffic flow data. Moreover, the data coming from the weather station is a complement. The traffic flow data are represented as the number of vehicles per 15-min at a particular freeway. These data are normalized to real numbers in the range of $[0, 1]$. The weather data consist of temperature, humidity, visibility, wind gust and speed, dew point, cloud layer height, and weather condition. All of these items are normalized to real numbers in the range of $[0, 1]$, except for the weather condition which is soft-data.

The output of the predictor is the k-step ahead of traffic flow prediction values. In our proposed architecture, the training outputs of both the predictors are identical. Specifically, the predictor model 2 predicts the traffic flow using data coming from the weather stations. In this work, the traffic flow and weather data are collected on the weekdays and weekends from August 1^{st}, 2013 to November 25^{th}, 2013. The data from August to October are used for the training and November for the testing.

3.6.2 Scenario Settings

To find the suitable settings of the DBN, decision fusion schemes, and parameters of the Dempster's fusion method, several scenarios are proposed. In the streams of data processing and prediction part, the traffic flow and weather data-sets auto- and cross-correlation are analyzed to select the pertinent features. Next, the selected features are applied to different sub-scenarios. These sub-scenarios are defined based on:

1. Traffic flow prediction model: Autoregressive Integrated Moving Average (ARIMA), ANN, and DBN.
2. Feature settings: original, de-trended, and weekend/weekday traffic flow data.
3. Data fusion scheme: FEI-DEO and DEI-DEO data fusion levels.
4. DBN parameter settings of the event-based traffic flow prediction.
5. Final decision fusion methods.

For the first scenario, the most appropriate model for traffic flow prediction is obtained by feeding ARIMA, ANN, DBN with the original traffic flow data. Based on the experiment results, the prediction model with the lowest performance indices is selected. These indices are defined in 3.6.3.

In the second scenario, the selected prediction model is tested using different feature settings to assess the impact of these settings on the prediction performances. Mainly, the de-trended and the weekend/weekday versions of the traffic flow data are investigated. The goal of the de-trending is to remove the fluctuation caused by the variation of hours and days of each week. This can be done by estimating the seasonal variation component and subtracting this component from the original traffic flow data [63], as follows:

- let $f_i \in \mathcal{R}^T$ denotes time-series data of the traffic flow at freeway i for T total number of time stamps.
- define the hour of day and day of week pair (h, d), where $h = 0, 1, ..., 23$ and $d = 1, ..., 7$. From these definitions, the following seasonal variation component is defined as the following

$$s_i(h, d) = \frac{\sum_{\{t|g(t)=(h,d)\}} f_i(t)}{|\{t|g(t) = (h, d)\}|} \tag{3.6}$$

where

$$\begin{aligned} s_i(h, d) &\triangleq \text{seasonal variation at hour } h \text{ and day } d \\ &\quad \text{for road i.} \\ f_i(t) &\triangleq \text{traffic flow of road } i \text{ at instant time } t. \\ g(t) &\triangleq \text{an indexing operator for } h \text{ and } d \\ &\quad \text{for a given time } t. \\ |A| &\text{ denotes the cardinality of set A.} \end{aligned}$$

- use the above definitions and equation to calculate the de-trended version $\delta f_i(t)$ of the traffic flow $f_i(t)$ of the freeway i:

$$\delta f_i(t) = f_i(t) - s_i(h,d), \quad \forall g(t) = (h,d) \tag{3.7}$$

As can be seen in Figure 3.7, traffic flow shows different patterns during the weekday and weekend. Therefore, two experiments with two groups of data: weekend and weekday traffic flow data, are conducted. These groups of data are trained separately using two different settings of the selected model. The outputs are then concatenated and compared to the ground truth using the performance indices.

In the third scenario, the traffic flow and weather data fusion at different levels are investigated. Essentially, the relevant weather data as additional information to the traffic flow prediction model are included based on the data fusion schemes. In the FEI-DEO scheme, the data fusion is performed at the feature level. This scheme is compared with our novel architecture based on the DEI-DEO scheme. In this architecture, the traffic flow and weather data are predicted separately using DBNs, illustrated in Figure 3.4. In this type of architecture, the k_1-step in the past traffic flow data are trained using k_2-step in the future traffic flow data as the output. Similarly, the current weather data are trained using k_2-step in the future traffic flow data as the output. Both of the data are trained using DBNs similar to the feature-level data fusion method. The result of each the predictors is then merged using data fusion techniques. The techniques fuse a set of decisions produced by each prediction model to obtain enhanced new traffic flow prediction. In the data fusion part of the architecture, several methods of data fusion are tested: weighted average, ANN, and DBN.

In the next scenario, the event-based data processing and prediction are investigated. First, after the traffic-related city events are extracted from the Twitter data, traffic flow prediction is conducted using DBN. To find the suitable number of hidden units and layers, 10-fold cross validation is performed in the training data. When the prediction results of both the streaming of data and event-based data processing units are obtained, the traffic clustering is conducted. Subsequently, the performance of the clustering is analyzed.

In the last scenario, the decision coming from streams of data and event-based data processing units are fused using DSET. In this scenario, the classical DRC and Dempster's rule of conditioning are applied in the decision process. The performances of both the rules are discussed.

3.6.3 Performance Indices

The performance of different scenarios is evaluated using two performance indices: Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). These indices are defined as:

$$\text{RMSE} = \left(\frac{1}{T} \sum_{t=1}^{T} (y_t - \hat{y}_t)^2 \right)^{\frac{1}{2}}$$
$$\text{MAE} = \frac{1}{T} \sum_{t=1}^{T} |y_t - \hat{y}_t| \tag{3.8}$$

where y_t and $\hat{y}_t$ are the actual and predicted traffic flow at time t respectively. These indices are used to measure the linear score that averages the error with the same weight and to measure the residuals by assigning larger weights to larger errors. These measures are represented by RMSE and MAE respectively. After the traffic flow prediction results are clustered, classification accuracy, precision, sensitivity, specificity, G-Mean, and F1-measure are used to analyze the results.

3.7 Analysis and Results

In this section, the capabilities of the proposed prediction architecture compared with other traffic flow prediction techniques are demonstrated. The results are presented according to the proposed scenarios.

3.7.1 Correlation Analysis

The following sections provide a detailed analysis to demonstrate the use of auto-correlation and cross-correlation for feature selection.

Auto-Correlation

To determine the number of previous steps of traffic flow that are incorporated in the input of the prediction models, the auto-correlation analysis is performed. The purpose of the analysis is to test if the previous k-step intervals of the traffic flow have any correlation

with the k-step ahead traffic flow. Figure 3.9 shows the auto-correlation results for both original (Figure 3.9(a)) and de-trended (Figure 3.9(b)) traffic flow data. For both the figures, each lag corresponds to 15 minutes time step. The height of the red lines represents the correlation between traffic flow at (t) and $(t-\text{time lag})$. The two blue lines correspond to the 95% confidence intervals of the correlation coefficients.

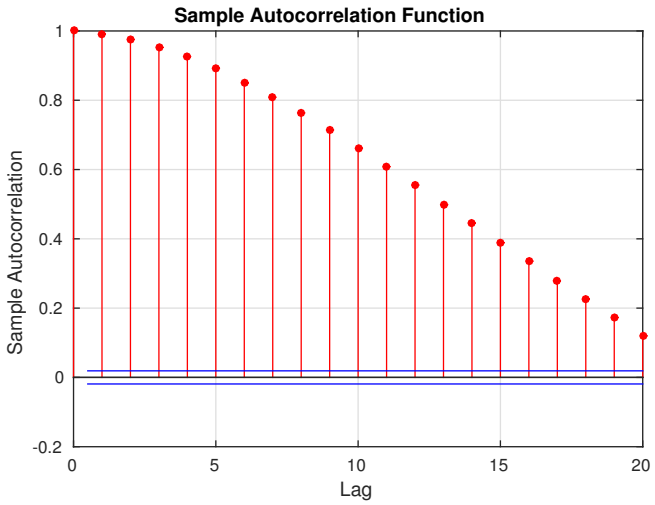
As can be observed in Figure 3.9 (a), even up to 20 lags (five hours) in the past, the original traffic flow values exhibit significant correlation. For the de-trended version, traffic flow values have significant correlation up to more than 20 lags, as depicted in Figure 3.9 (b). However, we limit the lag number to 20 in cross-validation processes due to computational constraint. In our cross-validation process, we fixed the predictor's parameters, and we train the predictor using different lag numbers. The outputs of the predictors are observed to determine the optimized lag number. The cross-validations are conducted only on the training set (August-October 2013). The experiment shows that the suitable lag numbers for both the original and de-trended version are eight.

Cross-Correlation

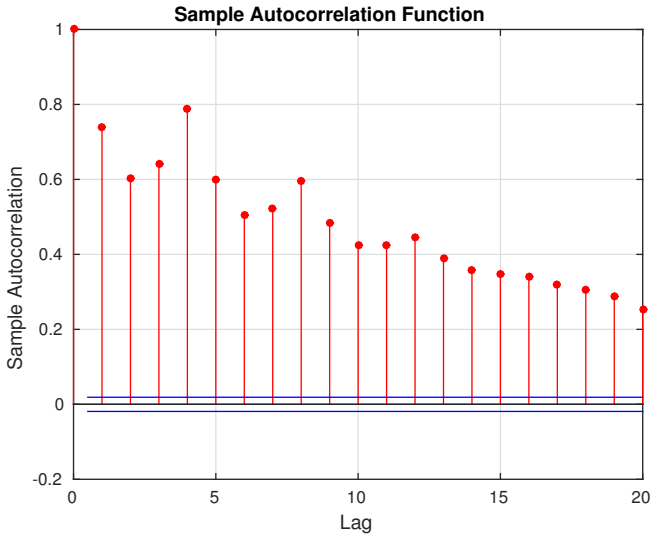
To properly select the most weather factors that shape traffic flow behavior, it is necessary to investigate the weather and traffic correlation. The cross-correlation results are depicted in Figure 3.10. The labels *Tr. Flow, Wt. Con, Temp, Hum, Vis, Wind. G, Dew. P, CHCH, and Wind S.* in Figure 3.10 (a) refer to *Traffic flow, Weather Condition (cloudy, sunny, snowy, etc), Temperature, Humidity, Visibility, Wind Gust, Dew Point, Cloud Height Coverage and Wind Speed* respectively.

In table 3.3, the cross-correlation coefficient between each pair of weather variables and traffic flow is represented by a real number. To select the pertinent feature subsets, the definition is given in [60] is followed. *Dew P,* has a small correlation coefficient with respect to traffic flow. Hence, this variable is discarded. In addition, Hum, CHC, and $WindS.$ are discarded since they have large correlation coefficients with respect to the highest cross-correlation coefficient, i.e., $Temp.$ Hum, CHC, and $WindS.$ are considered as redundant weather variables.

The final feature subset is plotted in Figure 3.10. As illustrated in the figure, $Temp$, $WindG.$ and $Wt.Con$ have a positive correlation with respect to traffic flow. Obviously, traffic volume increases when weather presents high temperatures (daytime) and decrease during nighttime. Furthermore, it can be noticed that traffic flow is inversely correlated to Vis (-0.12). Indeed, during low visibility condition such as fog, rain, etc, the average speed is lower than its normal level which yields to lower traffic flow. In addition, these inclement



(a) Original traffic flow data



(b) De-trended traffic flow data

Figure 3.9: Auto correlation of the traffic flow data.

Table 3.3: Traffic flow and weather variables cross-correlation coefficients

	Tr. Flow	Wt. Con	Temp	Hum	Vis	Wind G.	Dew P.	CHC	Wind S.
Tr. Flow	1.00	0.20	0.41	-0.32	-0.12	0.12	0.09	0.31	0.34
Wt. Con	0.20	1.00	0.08	0.21	-0.08	0.08	0.28	0.84	0.23
Temp	0.41	0.08	1.00	-0.52	0.18	0.15	0.46	0.18	0.56
Hum	-0.32	0.21	-0.52	1.00	-0.22	-0.14	0.43	0.00	-0.34
Vis	-0.12	-0.08	0.18	-0.22	1.00	0.04	-0.02	-0.08	0.10
Wind G.	0.12	0.08	0.15	-0.14	0.04	1.00	0.04	0.12	0.38
Dew P.	0.09	0.28	0.46	0.43	-0.02	0.04	1.00	0.19	0.25
CHC	0.31	0.84	0.18	0.00	-0.08	0.12	0.19	1.00	0.31
Wind S	0.34	0.23	0.56	-0.34	0.10	0.38	0.25	0.31	1.00

weather factors can affect the traffic flow such as traffic congestion. The conclusion made in this section vindicates the decision to use only pertinent weather data as part of the inputs in the traffic flow prediction architecture.

3.7.2 Traffic Flow Prediction and Data Fusion Analysis

Here, the proposed architecture are tested. Firstly, the suitable configuration and predictors' parameters of the streams of data processing unit are investigated. Furthermore, the settings of the predictor for the event-based data unit are configured. Finally, the final decision of both the units are fused and analyzed.

DBNs require several parameters that should be defined and tuned. For ARIMA, the number of regression variables are determined using auto-cross correlation discussed in the previous section. In the case of ANN, the number of hidden units and epoch are set through cross-validation on the training data. The hidden units are varied between $[50, 300]$ in the step of 10. Moreover, the number of epoch is varied between $[25, 250]$ in the step of 25. The best architecture for ANN is 1-hidden layer with 90 hidden units, and the number of epoch is equal to 150. In the case of DBN, the layer size from 2 to 5 layers are chosen. The number of nodes in each layer is chosen from $[50, 300]$ in the step of 50. The number of epochs is crucial to the learning phase. Therefore, the epochs range from 50 to 500 in the step of 50 are varied. The best architecture for DBN consists of 3 hidden layers, 250, 200, 100 hidden units in the first, second, and third hidden layers respectively, where the best number of epoch of the DBN training is found to be 100 epochs.

The DBN are compared to the most relevant state of the art prediction techniques namely The ARIMA and ANN. Figure 3.11 presents the error values of the DBN, as well

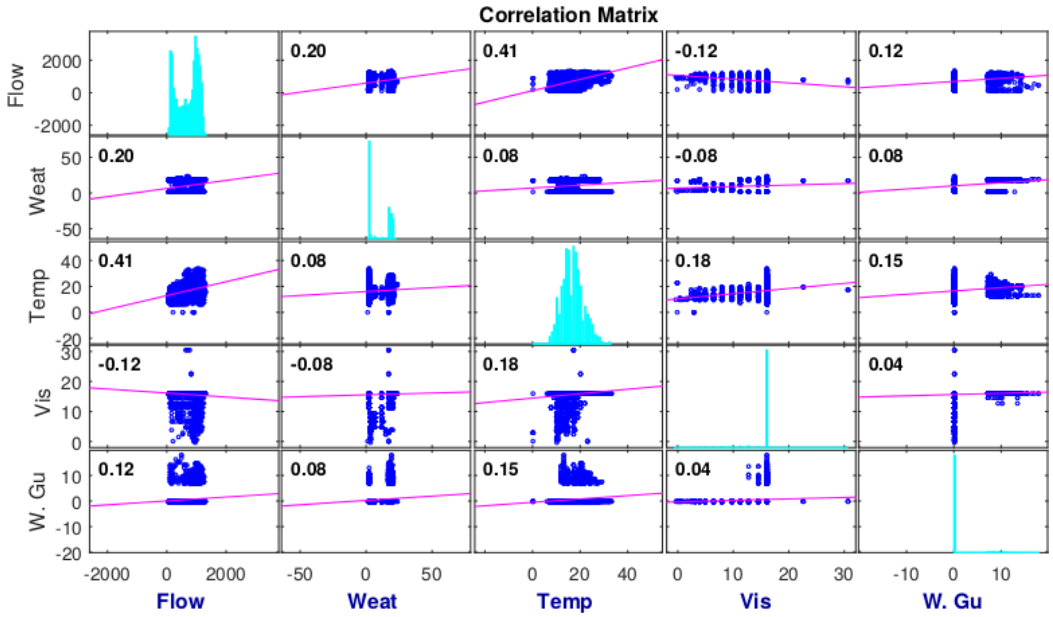


Figure 3.10: Cross-correlation between traffic flow and weather variables

as the ARIMA and ANN, in terms of RMSE and MAE. The same testing dataset for the three prediction approaches are used. It can be seen from the figure that for 15-min traffic flow prediction task, DBN exhibits the best performances. It presents an average MAE equal to 0.07 and an average RMSE equal to 0.05. Compared to the DBN, the ANN has lower performance (0.08 for MAE and 0.06 for RMSE) but higher performances than ARIMA. The results confirm the merit of DBN to predict the actual traffic flow. Hence, DBN stands out as a promising prediction technique to predict traffic in complex transportation network. DBN is selected to be the prediction model in our proposed architecture.

- Effect of de-trended and weekday/weekend version:

Transportation systems exhibit temporal fluctuation (daytime, nighttime), as can be seen in Figure 3.7. These fluctuations are removed by de-trending the traffic flow data. The impact of the de-trending on the DBN are explored. Three typical freeways, i.e., 17S, 87N and 15E, are chosen to represent low, medium, and high traffic flow. Furthermore, Several experiments are conducted and the performance based on MAE and RMSE are evaluated. The results are shown in Table 3.4.

The reported results show that using original traffic flow ensures better prediction

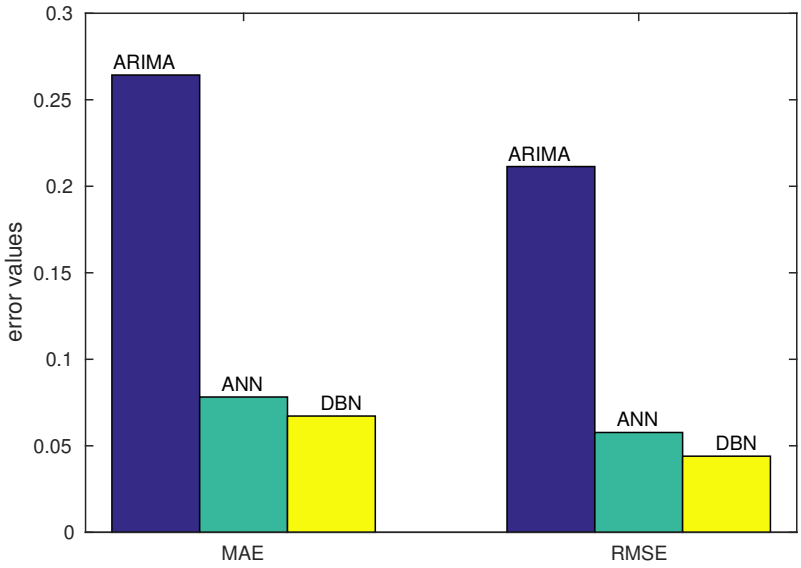


Figure 3.11: Errors comparison of DBN with other approaches

Table 3.4: Performance comparison of DBN using original data with the de-trended version and weekend-weekday partition

		Original	De-trended	Weekend/ Weekday
Low Traffic (17S)	MAE	**0.0410**	0.0635	0.1904
	RMSE	**0.0599**	0.0848	0.2441
Med Traffic (87N)	MAE	**0.0347**	0.0602	0.2166
	RMSE	**0.0455**	0.0916	0.2851
High Traffic (152E)	MAE	0.0850	**0.0541**	0.1980
	RMSE	0.1161	**0.0894**	0.2781
Average	MAE	**0.0487**	0.0677	0.2004
	RMSE	**0.0701**	0.1040	0.2659

Table 3.5: Performance comparison of DBNs

		Tr. Only	Weat. Only	T+W-DF1	T+W-DF2-LS	T+W-DF2-ANN	T+W-DF2-DBN
Low Traffic (17S)	MAE	0.0410	0.2188	0.0480	0.0455	0.0424	**0.0352**
	RMSE	0.0599	0.2722	0.0575	0.605	0.0580	**0.0481**
Med Traffic (87N)	MAE	0.0347	0.2282	0.0499	0.0342	0.0262	**0.0250**
	RMSE	0.0455	0.2921	0.0583	0.0444	0.0366	**0.0356**
High Traffic (152E)	MAE	0.0850	0.2209	0.0472	0.0853	0.0714	**0.0654**
	RMSE	0.1161	0.2754	0.1038	0.1150	0.1012	**0.0954**
Average	MAE	0.0487	0.2192	0.0416	0.0491	0.0433	**0.0405**
	RMSE	0.0701	0.2722	0.0656	0.0700	0.0638	**0.0603**

performances when the traffic flow is low and medium. Indeed, for medium traffic, the DBNs fed with original data have 0.034 and 0.045 as MAE and RMSE values respectively. Whereas, de-trended version presents 0.06 and 0.09 as MAE and RMSE values respectively. This results can be explained by the fact that the majority of the freeways has medium traffic flow. It is worth noting that feeding prediction model with de-trended traffic data outperforms when the traffic flow is high. Overall, the original traffic data outperforms de-trended version. This revelation can be explained by the fact that de-trended traffic data removes the temporal context of the features. However, with original data, the spatial and temporal relationship of the features are retained.

Moreover, the results show that the trained DBNs, when the weekday/weekend version is considered, present higher average error. For example, in the 87N freeway, the DBNs have an RMSE value equal to 0.0455 considering the whole traffic while weekday/weekend version shows an RMSE equal to 0.285. This high error value is due to the few quantity of data during the weekend. Secondly, traffic data is time-series data. Thus, analyzing weekday and weekend data separately hides the time correlation inside the data between weekday and weekend.

- Traffic and weather data fusion analysis:

As mentioned in [148], traffic flow is closely related to weather conditions. However, existing studies ignore these conditions when using DBN to predict traffic flow in transportation networks. In the following experiment, the performances of data fusion in different levels are investigated. To this end, two levels are differentiated: at feature level, which is denoted by $T+W-DF1$, and at decision level. Furthermore, at decision level, the performance of three variants of data fusion techniques are examined. Specifically, the traffic predictions coming from the two predictors are fused using:

- Least Square, which is denoted by $T+W-DF2-LS$.

Table 3.6: Final decision statistical measures

	Streams of Data	Event-Based Data	Classical DRC	Dempster's Conditional
Accuracy	86.73%	83.24%	84.44%	88.91%
Precision	82.25%	80.72%	80.31%	85.96%
Sensitivity	86.61%	83.50%	84.38%	89.89%
Specificity	84.01%	81.11%	82.12%	85.63%
G-Mean	84.12%	81.67%	82.61%	86.23%
F1-Meas	83.82%	81.17%	82.95%	86.33%

- ANN, which is denoted by $T + W - DF2 - ANN$.
- DBN, which is denoted by $T + W - DF2 - DBN$.

The results of average values of MAE and RMSE are summarized in Table 3.5. In this table, the first and second columns show the average error values of the traffic flow prediction based on only traffic and weather data respectively. The third column depicts the results of the data fusion at the feature level. In general, fusing the data at feature level exhibits better prediction performances than using traffic or weather data only as input for the predictor.

In $T + W - DF1$ model, the DBN parameters are tuned simultaneously for traffic and weather data. Thus, the degree of freedom to tune the parameters is limited. Therefore, this level of data fusion presents inferior performances compared to $T + W - DF2 - LS$, $T + W - DF2 - ANN$, and $T + W - DF2 - DBN$. From results depicted in Table 3.5, it is clear that using DBN for decision fusion significantly outperforms Least Square and ANN for different traffic flow levels (Low, medium and high). The merit of the DBN for decision fusion is more clear in medium traffic flow (87N freeway). Indeed, $T+W-DF2-DBN$ has MAE and RMSE equal to 0.0250 and 0.0356 respectively. Meanwhile, $T+W-DF2-ANN$ has MAE and RMSE equal to 0.0262 and 0.0366 respectively.

For illustration purpose, the profile of predicted traffic flow values using decision-level data fusion based on DBN versus the ground truth for the three representative freeways (17S, 87N and 15E) are plotted in Figure 3.12. From this figure, it can be seen that our proposed architecture enhanced with weather data better tracks the traffic flow pattern in medium traffic flow (87N freeway).

In the case of DBNs using traffic flow and Twitter data, the layer size from 2 to 5 layers are chosen. The number of nodes in each layer is chosen from $[50, 300]$ in the step of 50.

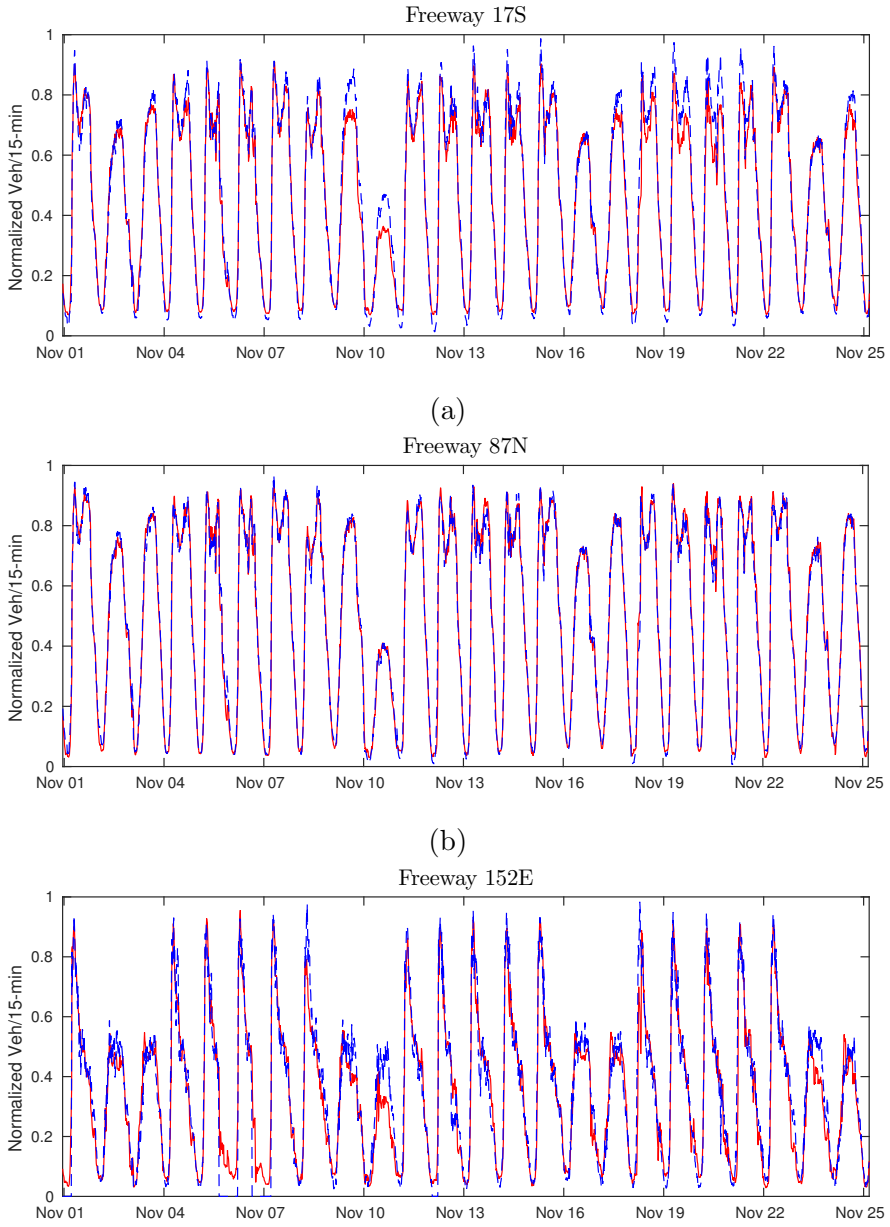


Figure 3.12: Traffic flow prediction of freeway with various traffic volume. Freeway with low (a), medium (b), and high (c) traffic flow.

The number of epochs is crucial to the learning phase. Therefore, the epochs range from 50 to 500 in the step of 50 are varied. The best architecture for DBNs consists of 3 hidden layers, 250, 225, 150 hidden units in the first, second, and third hidden layers respectively, where the best number of epoch of the DBNs training is found to be 150 epochs. After the prediction results of both the streams of data and event-based units are obtained, the next step is to cluster the traffic into four levels of intensity: *Low, Medium, High,* and *Very High*. From here, the masses of each level including the boundaries are calculated using the method proposed in Chapter 4. Finally, DSET is used to perform the final fusion of the decisions. Note that the Dempster's conditional updating is implemented with a static reliability parameter; the $\lambda(t) = \lambda$ is tuned using cross-validation until the best result is acquired. The results of the predictions individually and final decisions are presented in Table 3.6.

The table shows that when the classical DRC is used, the combination of both units is worse than those of the individual units. The DRC acts as an averaging method for both units. However, when the Dempster's conditional updating is employed, higher accuracy is achieved. This result is achieved when the parameter $1 - \lambda$ is very small. Although the improvement is not significant, the Dempster's conditional updating shows promising results.

3.8 Summary

This chapter presented the details of the big-data-based traffic flow prediction architecture. The main goal of the proposed architecture is to handle streams of data and event-based data in a given smart city. A novel six-layered traffic flow prediction for both the data is presented. Each layer contributes significantly to handling big-data-based traffic prediction problem. The details of for data preprocessing, feature selection, Twitter traffic-related city events extraction, as well as traffic clustering and mass assignment methods, are presented.

The experimental results of the proposed architecture are presented. The experiments start with the auto- and cross-correlation studies to find the pertinent features that affect most the traffic flow prediction. Next, the different settings for traffic flow data are investigated. Furthermore, the details of the parameters selection and model configuration are provided. The performance of each component in the proposed architecture is presented and used as a comparison to the overall prediction accuracy. Lastly, unreliable data handling using DSET is another capability that is added to the architecture, and help the overall proposed architecture to improve the prediction accuracy.

Chapter 4

Non-Stationary Traffic Flow Prediction[1]

4.1 Introduction

The stationarity assumption is especially appealing in time series analysis due to the widely available models, prediction methods, and well-established theory. However, applying this assumption to real-world data, which are mostly non-stationary, might lead to inappropriate forecasts. Traffic flow time series can be either stationary or non-stationary. The non-stationarity problem in traffic flow arises when irregular traffic flows due to scheduled or unexpected events such as accidents or constructions occur. In principle, long-term traffic flow predictions may not be accurate enough for reliable practical use if the data are assumed to be stationary. One of the solutions to handle non-stationary traffic flow is to consider the data as a collection of piece-wise, or locally, stationary data. This means the statistical properties of the time series are changing but remain constant for a specific period of time. In relation to prediction problems, using the piece-wise stationarity assumption, the stationary part of the time series can be learned using stationary methods. Subsequently, the predictor is updated when the time-series move to a different stationary

[1]This chapter contains an article published in 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Arief Koesdwiady, Safaa Bedawi, Chaojie Ou, and Fakhri Karray, "Non-Stationary Traffic Flow Prediction Using Deep Learning". The contents of this chapter have been incorporated within a paper that has been uploaded to ArXiv and is going to be submitted for publication. Arief Koesdwiady, Fakhri Karray, "SAFE: Spectral Evolution Analysis Feature Extraction for Non-Stationary Time Series Prediction".

state. It is therefore imperative to continuously check whether the time series is stationary or not.

In this work, a non-stationarity detection method, which can be used in real-time and combined with powerful predictors such as state-of-the-art machine learning techniques, is developed. Generally, there are two ways to update predictors: passive and active methods [40]. In the passive method, predictors adaptation is done regularly, independent of the occurrence of non-stationarity [84, 58, 44]. These methods are quite effective for prediction with gradual changes in the data. However, the main issues are the huge computational requirements and the potential for over-fitting. On the other hand, the active detection methods monitor the data to detect changes and perform adaptation only when it is needed. These can be done either by monitoring the error of the predictor [109] or monitoring the features of the data [28, 9, 97, 10]. The main issue of the first approach is that the error might not reflect the non-stationarity of the data and it heavily relies on the prediction accuracy of the predictor, which can be misleading if a poor training process is used to build the predictor. Therefore, an active detection mechanism based on the features of the data is considered in this work.

The proposed detection approach is developed by monitoring the evolution of the spectral contents of a time series. The main hypothesis is that frequency domain features contain more information than time domain ones do. Furthermore, the method is specifically developed to work in combination with state-of-the-art machine learning methods such as Deep Neural Networks (DNN). By combining the power of frequency domain features and the known generalization capability and scalability of DNN in handling real-world data, it is expected that high prediction performances can be achieved.

However, it is known that connectionist models are subjected to a serious problem known as *Catastrophic Forgetting*, i.e., forgetting the previously learned data when learning new data [49]. Researchers have been trying to deal with this problem by using ensemble learning methods [133, 122], evolutionary computing [47], and focusing on the regularization aspects of the models [61, 80]. These methods are mainly tested on classification problems. In a regression problem, more specifically real-world time series problem, it is highly possible that the patterns in the past might not appear again in the future, and the future data is highly affected by the past data that are close to the future only. Therefore, an approach to include some previous data in the past, which size is variable with respect to the degree of non-stationarity, is proposed.

The main contributions here are summarized as follows:

- The development of an algorithm to detect non-stationarity based on the evolution of the spectral contents of the traffic flow time series.

- The development of an online learning framework that combines the detection method with online machine learning methods efficiently.
- The development of an algorithm to proportionally include some data in the past to handle Catastrophic Forgetting.
- Comprehensive experiments to validate the main hypothesis and show the effectiveness of the proposed approach. Rigorous experiments to test different distance functions to monitor the evolution of spectral contents are performed.

The combination of the detection algorithm, online learning framework, and proportional algorithm is called Spectral Analysis Feature Extraction (SAFE). The rest of the chapter is organized as follows. Section 4.2 describes the main approach developed in this work. Section 4.3 explains the mechanism for embedding predictors with SAFE. Section 4.4 elaborates on the datasets, experimental settings, and performance metrics used to validate the hypothesis and show the effectiveness of the proposed framework. Section 4.5 presents the experimental results and discussions. Finally, section 4.6 concludes the chapter.

4.2 The SAFE Approach

In this section, the proposed SAFE approach is presented. SAFE is a technique for explicitly detecting non-stationarity in time series. This technique monitors the evolution of the local spectral energy of a time series and computes the discrepancy between the energy at present and previous time instances. The discrepancy then is used to test whether non-stationarity presents or not.

SAFE consists of two main modules: feature extraction, and non-stationarity detection modules. In the first module, Short-Time Fourier Transform (STFT) is applied to extract frequency contents of the time series at each instant time. The results of STFT are frequency values in the complex form. Therefore, the spectral energy of each frequency is computed to simplify the calculations.

The second module uses Simple Moving Average (SMA) and Exponentially-Weighted Moving Average (EWMA) [131] methods to estimate the long-term and immediate responses of the evolution of the spectral energy through a distance function. In other words, the difference of the spectral energy at every instant time is considered rather than the spectral energy itself. In an online learning setting, an incoming observation together with its past values are concatenated to form a window in which the STFT is performed.

The result of the transformation then compared with the previous window using a distance function. This way, changes can be detected as soon as a new observation arrives.

4.2.1 Frequency-Domain Feature Extraction

In the literature, several time-domain statistical features are used to characterize a time series [28, 9]. In this work, a frequency domain approach is presented. There are two main hypothesis in this work:

- In stationary conditions, the extracted features are expected to be stationary, or at least fluctuating around a stationary value. Therefore, whenever this is not the case, it can be deduced that a non-stationarity presents.
- More information can be gained in frequency domain than that in its counterpart. Therefore, it is expected that the non-stationary detection is more accurate, in terms of true positive and detection delay.

In the previous section, it is assumed that non-stationary time-series can be split into chunks of stationary parts. Therefore, a sliding window with a sufficient width can be applied to obtain local stationarity status of a signal. Therefore, it is intuitively suitable to apply STFT to extract the frequency contents of the signal. The discrete STFT can be expressed as

$$\mathrm{STFT}(m,\omega) = \sum_{k=-\infty}^{\infty} x[k]w[k-m]e^{2\pi j\omega k/L} \tag{4.1}$$

where the $x[k]$ is the time series of interest, m, and ω represent the indicators of time and frequency, respectively; while L is the length of the window function w. In this work, a hamming window function is used. The choice of the sliding window width determines the time-frequency resolution. A wide window provides better frequency and blurred time resolutions while a narrow window works in an inverse way. Once the the complex values of each frequency of interest are computed, their spectral energies are then computed. Figure 4.1 illustrates the process of STFT for every sliding window. Take $\mathrm{STFT}(t+3,\omega)$ as an example. When a new observation $x(t+3)$ arrives, the STFT is computed using this values and its several values in the past. It is expected that $\mathrm{STFT}(t+3,\omega)$ and $(t+2,\omega)$ will be similar if the series is stationary and diverged if it is not the case.

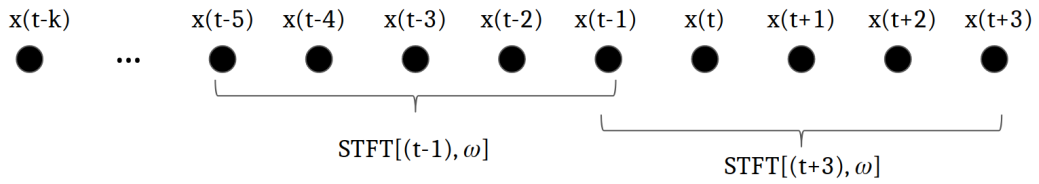


Figure 4.1: An illustration of the STFT process.

To show the effectiveness of the frequency-domain feature extraction on capturing the non-stationarity, a small experiment is conducted and the results are shown in Figure 4.2. In this experiment, four modes of non-stationarity are injected to the time series:

- The first point, which is denoted by (1) at $t = 300$, illustrates a gradual non-stationarity in term of variance.
- The second point, which is denoted by (2) at $t = 600$, shows an abrupt non-stationarity in term of the mean of the series. After this point, the mean is constant, which introduces a bias in the series.
- The third point, which is denoted by (3) at $t = 900$, depicts an abrupt non-stationarity in term of the mean of the series. However, the mean keeps increasing after this point. At this interval, the non-stationarity is in continuous mode.
- In the last point, the series goes back to the original stationary form.

Indeed, there are many modes of non-stationarity that are not included in the experiment. However, these modes are sufficient to illustrate necessary behavior of non-stationarity for time series prediction. The figure also shows that the spectral energies represent the behavior of the process, which in this case is concentrated in the lower frequency bin. The energy behavior after point (2) looks similar to the one before point(1) although they have different means. This should not be considered as a problem since the important part is the changes at the point of interest, which will be reflected when the distance between points is calculated. It should also be noted that the last point, when the system goes back to the original form, is important to consider since in connectionist modeling the predictor tends to forget the past when a new concept is learned. This creates problem called *Catastrophic Forgetting*. By keep monitoring the changes, the predictor can be trained to learn previous concept when necessary.

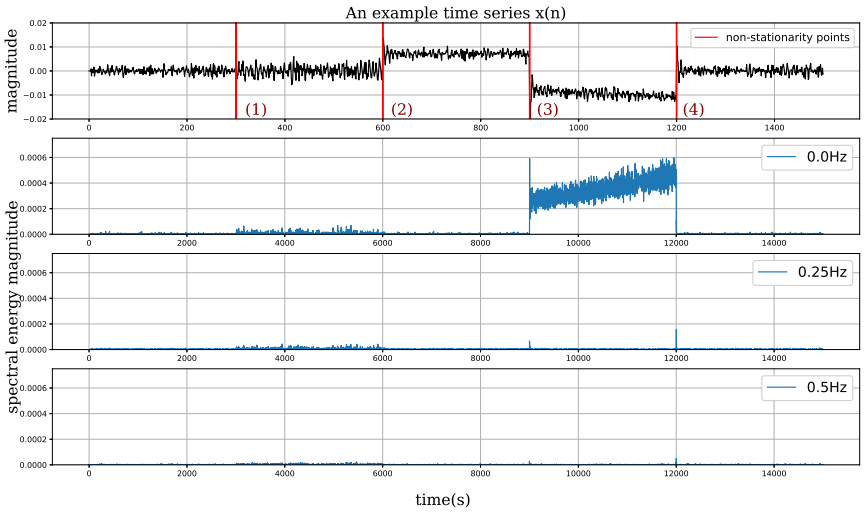


Figure 4.2: An example time series (top) with its spectral energy contents (the rest).

4.2.2 Non-stationarity Detection Module

The next step after extracting features is to detect whether a non-stationarity occurs or not using the non-stationarity detection module. There are two sub-modules in this module: the distance module, which computes the similarity between two consecutive extracted features, and the non-stationarity test module, which decides whether the observed distances translate to non-stationarity or not. Furthermore, to find the most compatible distance function that can capture the non-stationarity better, three distance functions are tested, namely the Euclidean, absolute Cosine, and absolute Pearson distances. The absolute term is needed since we are interested only in the magnitude of similarity, and not in the direction.

The subsequent step is detecting non-stationarity based on the computed distances, which is developed based on SMA and EWMA. EWMA method estimates the moving mean of a random sequence by giving more importance to recent data. For a set of values of a variable given as $X = \{x_1, x_2, \cdots, x_n\}$, which has mean μ_0, the EWMA estimates the following

$$Z(t) = (1 - \lambda)Z(t-1) + \lambda x(t), \quad t > 0 \tag{4.2}$$

where $Z_0 = \mu_0$. The parameter λ weighs the importance of the recent data compared to the older ones. This is suitable to the proposed feature extraction method, where a

new observation is appended to the sliding window and, in the same time, should provide immediate insight that a non-stationarity occurs. In addition, this capability is especially important for detecting gradual non-stationarity. Furthermore, the standard deviation of $Z(t)$ is given as

$$\sigma_z(t) = \sqrt{\frac{\lambda}{2-\lambda}(1-(1-\lambda)^{2t})\sigma_x} \tag{4.3}$$

The output of EWMA represents the immediate estimated mean of the distance while SMA represents the long-term evolution of the mean. Based on this output, a decision about nonstationarity has to be made. To do this, an algorithm called SAFE is proposed. This algorithm is illustrated in Algorithm 5. The input of the algorithm is a sequence of data or time series, and initialization is required for some parameters such as λ for the weight of EWMA; $w = 0$ for the warning counter; T for the trigger detection threshold; W for the warning detection threshold; and γ for the warning duration.

The first step in the algorithm after initialization is to append new incoming data to the window of previous data to form x_{temp}, as depicted in line 2 of the algorithm. Next, STFT is applied to x_{temp}, which results in $f(t)$. Subsequently, the distance $d(t$ of $f(t)$ with its previous $f(t-1)$ is computed. In the initial stage of the algorithm, $[x(t-k), \cdots, x(t-1)]$ and $f(t-1)$ are not available. It is safe to assume that $[x(t-k), \cdots, x(t-1)] = [0, 0, \cdots, 0]$ and $f(t-1) = f(t)$ since it is not going to significantly affect the rest of the computation.

The next step is to apply both EWMA and SMA to $d(t)$, which results in $Z(t)$ and $\bar{\mu}(t)$. The $\bar{\mu}(t)$ is necessary since it represents the long-term states of $d(t)$, in particular when a non-stationarity is continuously occurring, as depicted Figure 4.2 point 3 to 4. Furthermore, the standard deviation of $Z(t)$ is calculated using Equation 4.3. This standard deviation is used together with the control limits W and T as moving thresholds to determine whether $Z(t)$ is still inside a particular stationary mode or not. This idea is illustrated in Figure 4.3. Line 8 and 11 of Algorithm 5 impose the moving thresholds to $Z(t)$. If at an instant time $Z(t)$ is greater than $\bar{\mu}(t) + T * \sigma(t)$, then the non-stationary flag $ns(t)$ is raised. However, when $Z(t)$ is greater than $\bar{\mu}(t) + W * \sigma(t)$, the algorithm waits until certain duration γ to raise the flag. This is also useful when $Z(t)$ is fluctuating insignificantly, which might be due to outliers and/or other unpredictable factors in the data. The flag can be used by predictors to update their parameters when necessary, which is more efficient compared to the blind adaptation scheme.

Algorithm 5 SAFE algorithm.

Input: sequence of data
Initialize: λ, $w = 0$, T, W, γ

for every instant t, a new data $x(t)$ arrives **do**
  $x_{temp} = [x(t-k), \cdots, x(t-1), x(t)]$
  $f(t) = STFT(x_{temp})$
  $d(t) = dist_{func}(f(t-1), f(t))$
  $Z(t) = (1-\lambda)Z(t-1) + \lambda d(t)$
  compute the SMA of a sufficiently long sliding window of $d(t)$, this is denoted by $\bar{\mu}(t)$.
  compute $\sigma(t)$ according to Equation 4.3.
  if $Z(t) \geq \bar{\mu}(t) + T * \sigma(t)$ **then**
    $ns(t) = 1$
    $w = 0$
  else if $Z(t) \geq \bar{\mu}(t) + W * \sigma(t)$ **then**
    $w = w + 1$
    if $w \geq \gamma$ **then**
      $ns(t) = 1$
      $w = 0$
    end if
  else if $Z(t) \leq \bar{\mu}(t) + W * \sigma(t)$ **then**
    $w = \max(0, w - 1)$
  end if
end for

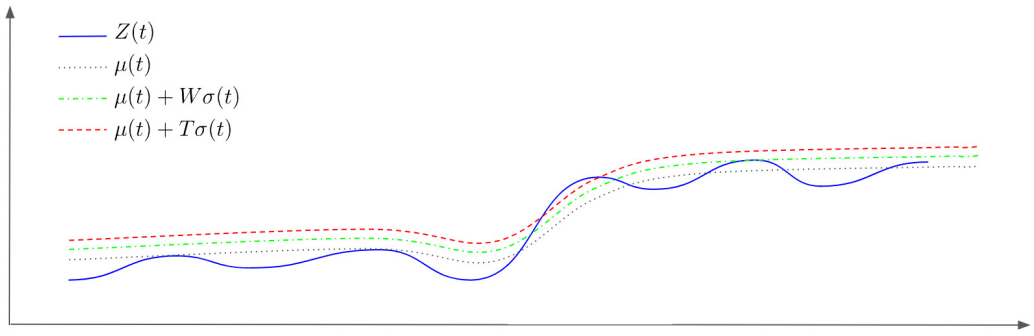


Figure 4.3: Illustration of the SAFE approach.

4.3 Embedding SAFE to Online Predictors

An online predictor is applied only when a non-stationarity is detected. Initially, a predictor is trained using a presumed stationary data set in an off-line manner. The parameters obtained from this training are used as initial conditions. In a simulation case, a period of data where the stationary properties hold can be selected; while in a real-world case, an initial predictor is trained using all data available at hand.

Once the non-stationarity has been detected, the next step is to update the parameters of the chosen predictor. The predictor should support online learning since some predictors require training from scratch when new data are available. Some notable online learning algorithms are online passive-aggressive [36], linear support vector machine (SVM) with stochastic gradient descent (SGD) [22], and deep neural networks. These learning algorithms are suitable to combine with SAFE. Furthermore, mini-batch SGD also will be more suitable for SAFE since we can include previous data points to form a mini batch and update the predictors accordingly.

It is known that updating these predictors leads to catastrophic forgetting. To avoid this problem we include previous data so that the model also learn the new data using a portion of the data from the past. The question is, how many data points should we include in the online adaptation? To answer this question, the proportional deviation algorithm is introduced. The main idea of this algorithm is to include several numbers of data points proportional to the absolute deviation of $Z(t)$ from $\bar{\mu}$. Large deviation means the new data is far from the previous stationary point. Therefore, it is intuitively suitable to include more data when the deviation is large and vice versa. The size of the mini batch is computed as follows

$$u = \text{round}(\beta |Z(t) - \bar{\mu}(t)|) \tag{4.4}$$

where u is the number of data points included in the past or the size of the mini batch, and β is the proportional gain to scale the mini batch. The choice of β depends on the applications and affects the speed of the adaptation. The *round* operation rounds the calculation to the nearest integer. This operation is necessary since the size of the mini batch has to be an integer. Algorithm 6 is introduced to illustrate this procedure.

In this algorithm, the predictor is trained using $\{x_{train}, y_{train}\}$, and validated using $\{x_{val}, y_{val}\}$. The validation is used to control the training epoch. When the validation error converges, then the training is stopped. This algorithm can be considered as the sub-algorithm of Algorithm 5.

Algorithm 6 Embedding SAFE to predictors.

Input: $ns(t)$, $Z(t)$, $\bar{\mu}(t)$
Initialize: β

for every instant t **do**
  if $ns(t) == 1$ **then**
    $u = \text{round}(\beta\ |Z(t) - \bar{\mu}(t)|)$
    $x_{train} = [x(t-u), \cdots, x(t-1)]$
    $y_{train} = [y(t-u), \cdots, y(t-1)]$
    $x_{val} = x(t)$
    $y_{val} = y(t)$
    Train/update the chosen predictor using $\{x_{train}, y_{train}\}$ and validate the model using $\{x_{val}, y_{val}\}$.
  end if
end for

4.4 Experimental Settings and Datasets

In this section, the datasets and experimental settings that are used to test the hypothesis and to illustrate the effectiveness of the proposed approach are introduced. The data sets consist of two types: artificial and real-world data.

The artificial data allow an effective analysis to be performed since the exact locations of non-stationarity are known, which are not the case in real-world data. However, unless all possible non-stationary conditions can be captured, approaches to handle non-stationarity, which are tested using artificial data, might not work well in practice. Therefore, real-world data to test the effectiveness our approach are used. Since the exact locations and behavior of the non-stationarity is not known, the real-world data are tested in the prediction stages only. The goal of the test is to see whether the proposed approach is able to correct the prediction that is harmed by non-stationarity.

Several experimental settings to test each element in our approach are proposed. The settings are defined as follows:

1. Experiment to find the most suitable distance function for our approach. Three types of distance functions are investigated: Euclidean, Pearson, and Cosine distances. This is done using artificial data.

2. Experiment to test whether extracting features in frequency domain leads to better

non-stationarity detection results than in time domain. This is done using artificial data.

3. Experiment to investigate which predictor performs well in the proposed approach. This is done using linear and nonlinear artificial data.

4. Experiment to test which domain of feature extraction is better for prediction using real-world data.

At the end of this section, performance metrics used in each experiment are introduced.

4.4.1 Datasets

Here, the datasets used in the experiments are described.

Artificial Datasets

Two sets of artificial data are used in experimental settings 1, 2, and 3. The first set contains five time-series datasets that illustrate some behavior of stationary and non-stationary time series. These data sets are inspired, with modifications, by [34] and [85], and are given by the following processes:

(TS-A) Stationary AR(1) process. This is a stationary process with $\epsilon_t \sim \mathcal{N}(0, 1)$, and a range of values of $\alpha = \{0.7, 0.4, 0.1, -0.1, -0.4, -0.7\}$.

$$x_t = \alpha x_{t-1} + \epsilon_t, \forall 1 \leq t \leq T \tag{4.5}$$

(TS-B) Piece-wise stationary AR process with obvious changes. The changes in this process occur at two known locations $t_1 = 400, t_2 = 700$.

$$x_t = \begin{cases} 0.9x_{t-1} + \epsilon_t, & \text{if } 1 \leq t \leq t_1, \\ 1.68x_{t-1} - 0.81x_{t-2} + \epsilon_t, & \text{if } t_1 < t \leq t_2, \\ 1.32x_{t-1} - 0.91x_{t-2} + \epsilon_t, & \text{if } t_2 < t \leq T \end{cases} \tag{4.6}$$

(TS-C) Piece-wise stationary AR process with less obvious changes. The change in this process is less observable, and occur at $t_3 = 600$.

$$x_t = \begin{cases} 0.4x_{t-1} + \epsilon_t, & \text{if } 1 \leq t \leq t_3, \\ -0.6x_{t-1} + \epsilon_t, & \text{if } t_3 < t \leq T \end{cases} \tag{4.7}$$

(TS-D) Piece-wise stationary near-unit-root process with changing variance. This process has changes in variance occur at two points $t_4 = 400$, $t_5 = 750$, where $\epsilon_{2t} \sim \mathcal{N}(0, 1.5^2), \epsilon_{3t} \sim \mathcal{N}(0, 3^2)$.

$$x_t = \begin{cases} 0.999x_{t-1} + \epsilon_t, & \text{if } 1 \leq t \leq t_4 \\ 0.999x_{t-1} + \epsilon_{2t}, & \text{if } t_4 < t \leq t_5 \\ 0.999x_{t-1} + \epsilon_{3t}, & \text{if } t_5 < t \leq T \end{cases} \tag{4.8}$$

(TS-E) Piece-wise stationary ARMA(1, 1) process. This process has three points of changes $t_6 = 250, t_7 = 500, t_8 = 750$.

$$x_t = \begin{cases} 0.9x_{t-1} + \epsilon_t - 0.5\epsilon_{t-1}, & \text{if } 1 \leq t \leq t_6 \\ 0.3x_{t-1} + \epsilon_t, & \text{if } t_6 < t \leq t_7 \\ -0.7x_{t-1} + \epsilon_t + 0.6\epsilon_{t-1}, & \text{if } t_7 < t \leq t_8 \\ 0.4x_{t-1} + \epsilon_t - 0.1\epsilon_{t-1}, & \text{if } t_8 < t \leq T \end{cases} \tag{4.9}$$

The second set of artificial data is inspired by [28]. This set consists of two linear time series and two non-linear time series; both linear and non-linear time series are known to have parameter changes at some specified points. The linear datasets have a similar structure, which is given by AR(p) process. Time series linear-1 and linear-2 are respectively given by AR(4) and AR(5) processes. AR(p) process is given as follows:

$$x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \cdots + \alpha_p x_{t-p} + e_t \tag{4.10}$$

where $e_t \sim \mathcal{N}(0, \sigma^2)$.

The nonlinear time series are constructed using the following processes:

$$\begin{aligned} x_t =& [\alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \alpha_3 x_{t-3} + \alpha_4 x_{t-4}] \\ & * (1 - exp(-10x_{t-1})^{-1} + \epsilon_t) \end{aligned} \tag{4.11}$$

$$\begin{aligned} x_t =& \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + [\alpha_3 x_{t-1} + \alpha_4 x_{t-2}] \\ & * (1 - exp(-10x_{t-1})^{-1} + \epsilon_t) \end{aligned} \tag{4.12}$$

where Equation 4.11 and Equation 4.12 represent nonlinear-1 and nonlinear-2 time series, respectively. Table 4.1 presents the parameters used in all the linear and nonlinear time series. Column *Point* provides points where the parameters are implemented.

Table 4.1: Linear and Nonlinear Time Series Parameters.

Time Series	Point	α	σ^2
Linear-1	1-3000	$\{0.9, -0.2, 0.8, -0.5\}$	0.5
	3001-6000	$\{-0.3, 1.4, 0.4, -0.5\}$	1.5
	6001-9000	$\{1.5, -0.4, -0.3, 0.2\}$	2.5
	9001-12000	$\{-0.1, 1.4, 0.4, -0.7\}$	3.5
Linear-2	1-3000	$\{1.1, -0.6, 0.8, -0.5, -0.1, 0.3\}$	0.5
	3001-6000	$\{-0.1, 1.2, 0.4, 0.3, -0.2, -0.6\}$	1.5
	6001-9000	$\{1.2, -0.4, -0.3, 0.7, -0.6, 0.4\}$	2.5
	9001-12000	$\{-0.1, 1.1, 0.5, 0.2, -0.2, -0.5\}$	3.5
Nonlinear-1	1-3000	$\{0.9, -0.2, 0.8, -0.5\}$	0.5
	3001-6000	$\{-0.3, 1.4, 0.4, -0.5\}$	1.5
	6001-9000	$\{1.5, -0.4, -0.3, 0.2\}$	2.5
	9001-12000	$\{-0.1, 1.4, 0.4, -0.7\}$	3.5
Nonlinear-2	1-3000	$\{-0.5, 0.8, -0.2, 0.9\}$	0.5
	3001-6000	$\{-0.5, 0.4, 1.4, -0.3\}$	1.5
	6001-9000	$\{0.2, -0.3, -0.4, 1.5\}$	2.5
	9001-12000	$\{-0.7, 0.4, 1.4, -0.1\}$	3.5

Real-World Datasets

In this work, the traffic flow dataset downloaded from the Caltrans Performance Measurements Systems (PeMS) [1] is used. The original traffic flow was sampled every 30 seconds. These data were aggregated into 5-min duration by PeMS. Highway Capacity Manual 2010 [102] recommends to aggregate the data further into 15-min duration. We collected the traffic flow data of a freeway from January 1st 2011 to December 31st 2012. The dataset is split into three parts: from January 1st 2011-from August 31st 2011 for training, September 1st 2011-December 31st 2011 for validation, and the rest for testing.

4.4.2 Experimental Settings

The datasets are applied to four different experimental settings. In the first setting, TS-B and TS-C are used to find which distance function gives the best performance on clearly and not so clearly observable changes. In this part, the evolution of each distance function in relation to the time series is observed. Each dataset is tested over 100 trials to account for the randomness introduced by the noise. To gain insight on the performance, the total number of detection, false alarm, hit rate, missed detection, specificity, detection delay, and execution time are recorded.

The second experimental setting is performed to validate the hypothesis that more information can be gained by extracting the features in the frequency domain than in the time domain. TS-A is used to measure the false alarm, and TS-B to TS-E to measure the total number of detection, false alarm, hit rate, missed detection, specificity, detection delay, and execution time. In the experiment, each dataset is tested over 100 trials.

For the third setting, linear (Linear-1, Linear-2) and nonlinear (Nonlinear-1, Nonlinear-2) time series with changes are used to test the prediction performance when SAFE is embedded into three different predictors namely Passive-Aggressive Regressors, Kernel Linear-SVM, and Deep Neural Networks. In the Kernel Linear-SVM, the original features are mapped into a randomized low-dimensional feature space, which are then trained using linear SVM [124]. In this experiment, the mean-squared error (MSE), execution time, and percentage update required are measured. Furthermore, to account for the randomness, 20 simulations are conducted.

Finally, real-world datasets, namely Traffic Flow dataset, are used to measure the performance of deep neural networks which are combined with time-domain feature extraction and SAFE. In this experiment, the MSE, execution time, and percentage update required

are measured. In addition, all the experimental settings performance with graphs showing their responses over time are used for illustration.

4.4.3 Evaluation metrics

In this section, the performance metrics used in the experiments are described.

Total number of detection. This denotes the number of points detected in a time-series. The change point is considered true if it within 5% of the sample size. As an example, number of detection = 1 means there is only 1 change point is detected. This number is then cumulated over 100 trials. False positive, true positive, true negative, and false negative are denoted as FP, TP, TN as FN, respectively.

False Alarm Rate. It measures the number of change points detected that when there is no actual changes occur. Another name for this metric is False Positive Rate, which is given as $FP/(FP + TN)$.

Hit Rate. It measures the proportion of correctly detected changes over the actual number of change points, which is given as $TP/(TP + FN)$.

Missed Detection Rate. It is the number of undetected changes when there are actual change points over the actual number of change points, which is given as $1-$Hit Rate.

Specificity. It reflects the number of stationary points classified as stationary points over the actual number of stationary points. This is calculated as $TN/(TN + FP)$

Detection Delay. It measures the distance, in term of number of steps, of the detected changes from the actual points.

Execution Time. It measures the average time, in seconds, required to perform an experiment over a specific number of trials.

Mean-Squared Error. This metric measures the similarity between predictions and actual time series. Two types of MSE are used: the trajectory of MSE, which is defined as the evolution of the MSE at every instant time, and the overall MSE, which measures the total MSE of the whole test dataset.

Percentage of Update. The number of updates performed over the number of possible updates. As an example, a time series with 100 data points. The online update procedure is started from $t = 1$. If it is assumed that the algorithm updates the predictors 15 times. It means the percentage of the update is 15%. If blind adaptation scheme is performed, the percentage of the update will be 100%.

4.5 Results and Discussion

In this section, the experimental results to evaluate the performances of the proposed approach are discussed. The results are grouped according to the experimental settings mentioned in the previous section.

4.5.1 Distance Functions

There are various ways to compute the distance between two vectors; the popular ones are Euclidean, Pearson, and Cosine distances. To decide which distance function gives better performance on non-stationary detection, we conducted experiments using TS-B and TS-C datasets. These datasets illustrate the obvious and non-obvious changes that may occur in a time series.

First, it is important to show that these three distance functions can capture changes occur in a time series. Therefore, a simulation result showing the responses of the distances to the time series are presented. In this simulation, the SAFE approach is applied to the TS-C dataset with a sliding window of 5 samples. The TS-C dataset is chosen because it contains non-obvious changes. Intuitively, when more obvious changes occur, the response of the distance will be more distinguishable. Furthermore, the distances are calculated using the three distance functions mentioned in the previous paragraph. The result of this experiment is shown in Figure 4.4. This figure illustrates that when a non-obvious change occurs at a breakpoint $t = 600$, the three distances respond to the change immediately. It should be noted that the responses of these three distances are different, especially when using the Euclidean distance. Although Pearson and Cosine distances yield similar responses, they are still distinguishable. The response of the Pearson distance is a little bit smoother than that of the Cosine distance.

Next, the detection performances of the SAFE when it is combined with three distance functions are investigated. The size of the sliding window is kept at 5 samples. Other parameters that have to be set is λ, T, W and γ. It is suggested in [131] that $\lambda \in [0.1, 0.3]$. Since the detection performances highly depend on the thresholds T, W, these thresholds have to be set so that three distances have equal performance in one of the evaluation metrics. To achieve this, the thresholds are set such that the three distances have similar false alarm rate on TS-B, which is ≈ 0.05. From here, the other performance metrics can be judged given the false alarm rate.

To achieve the targeted false alarm rate, several trials of experiments are needed. Several values of λ are tested, and it is found that $\lambda = 0.3$ works best for all the three distances.

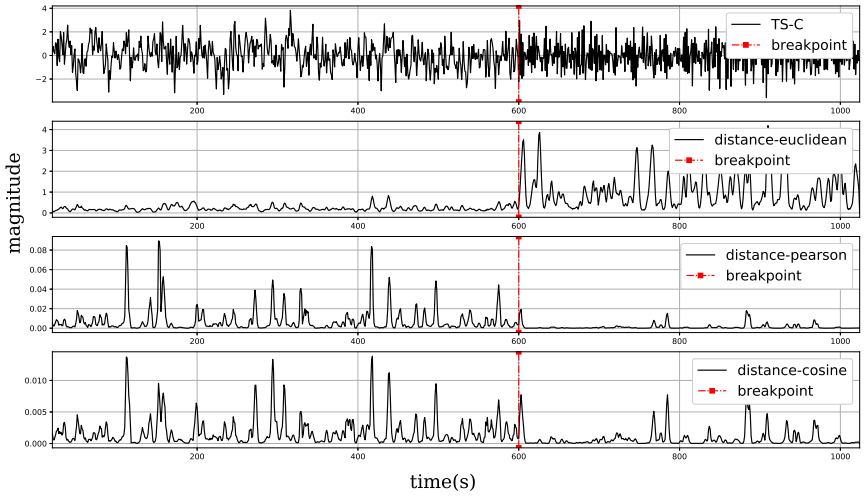


Figure 4.4: Response of distances to the TS-C breakpoint.

The warning thresholds for Euclidean, Pearson, and Cosine distances are 2.85, 0.75 and 1.4, respectively, and the trigger thresholds are 3.35, 1,25 and 1.9, respectively. The size of the sliding window for the SMA is 20; this value is able to capture the long-term evolution of the distances. It is expected that these parameters work for different types of non-stationarity. Therefore, these parameters are fixed for all experiments using TS-A to TS-E.

Table 4.2 shows the detection performances of the three distances over 100 simulations. It can be seen that in TS-B and TS-C datasets, Euclidean distance provides superior results in terms of the number of detection, hit rate, missed detection, and execution time. The highest average detection delay on TS-B is produced by Euclidean distance, but this is not the case on TS-C. Overall, Euclidean distance produces better performances than other distances do. Therefore, Euclidean distance is chosen as the distance function for the rest of the experiments.

4.5.2 SAFE Vs Time-domain Feature Extraction

In the second setting, the performances of SAFE and those of time-domain feature extraction (FE) are compared. The time-domain FE is inspired from the work in [28]. Four linear (auto-correlation, variance, skewness coefficient, kurtosis coefficient) and 1 nonlin-

Table 4.2: Summary of Non-stationary detection performance using TS-B and TS-C: Euclidean Vs Pearson Vs Cosine distance. Results over 100 simulations.

dataset / Perf.	TS-B		
	euclidean	pearson	cosine
# detection:			
0	0	3	0
1	4	30	5
2	96	67	95
False Alarm	0.049	0.050	0.050
Hit rate	0.98	0.82	0.97
Missed detection	0.02	0.18	0.03
Specificity	0.95	0.95	0.95
Det. delay (samples)	11.48 ± 11.62	8.59 ± 8.31	8.23 ± 7.45
Exec. time (s)	0.19 ± 0.004	0.26 ± 0.008	0.21 ± 0.003
dataset / Perf.	TS-C		
	euclidean	pearson	cosine
# detection:			
0	0	18	4
1	100	82	96
2	-	-	-
False Alarm	0.051	0.044	0.063
Hit rate	1.00	0.82	0.96
Missed detection	0.00	0.12	0.04
Specificity	0.95	0.96	0.95
Det. delay (samples)	6.9 ± 6.48	8.73 ± 9.46	7.83 ± 9.08
Exec. time (s)	0.19 ± 0.003	0.24 ± 0.007	0.19 ± 0.005

Table 4.3: Summary of false detection using TS-A. Results over 100 simulations.

α	False alarms	
	SAFE	time-domain FE
0.7	0.001	0.008
0.4	0.004	0.015
0.1	0.011	0.023
-0.1	0.019	0.030
-0.4	0.035	0.041
-0.7	0.052	0.053
Average	0.020±0.018	0.028±0.015

ear (bi-correlation) are extracted as the time domain features. The detection method after computing the distance is similar to the one used by SAFE in Algorithm 5.

The parameters of the SAFE approach is kept the same as the previous experimental setting, and the parameters of the time-domain FE are set to achieve around 0.05 false alarm rate on TS-B. To achieve this false alarm rate, the forgetting factor λ is set to 0.3 while T and W are set to 3 and 3.5, respectively. The sliding window size of the SMA is set to 20. Both SAFE and time-domain FE are experimented using TS-(A-E).

The first experiment on this setting is performed on TS-A. Since TS-A is a stationary time series, it is expected that low false alarm rates are found on both SAFE and time-domain FE. Table 4.3 shows that acceptable false alarm rates are found in both cases. It is evident that using different α SAFE produces lower false alarm rates than time-domain FE does. Indeed, the false alarm rate on stationary time series ideally should be 0. This can be achieved if both the thresholds are set to higher values. However, setting higher thresholds may lead to poor detection performances. This is up to the designer to decide which performances are important in their applications.

The second set of the experiments is done to test the non-stationarity detection performances using TS-(B-E). The results are summarized in Table 4.4. This table shows that the overall performance of SAFE on the datasets a better in terms of number detection, false alarm rate, hit rate, missed detection, specificity, detection delay, and execution time. It should be noted that although in some aspects time-domain FE has almost similar performances than SAFE does, time-domain FE has significantly higher average time to execute the experiments. In conclusion, time-domain FE executes the experiments more than 3 times slower than SAFE does.

To illustrate the performance of SAFE in more details, several graphs showing the

Table 4.4: Summary of Non-stationary detection performance using TS-(B-E): Frequency Vs Time domain. Results over 100 simulations.

dataset / Perf.	TS-B		TS-C	
	SAFE	time-domain FE	SAFE	time-domain FE
# detection:				
0	0	0	0	0
1	4	14	100	100
2	96	86	-	-
3	-	-	-	-
False Alarm	0.049	0.049	0.051	0.050
Hit rate	0.98	0.96	1.00	1.00
Missed detection	0.02	0.04	0.00	0.00
Specificity	0.95	0.95	0.95	0.95
Det. delay (samples)	11.48 ± 11.62	12.08 ± 11.50	6.9 ± 6.48	7.17 ± 6.54
Exec. time (s)	0.19 ± 0.004	0.72 ± 0.013	0.19 ± 0.003	0.73 ± 0.011
dataset / Perf.	TS-D		TS-E	
	SAFE	time-domain FE	SAFE	time-domain FE
# detection:				
0	0	0	0	0
1	8	15	3	1
2	92	85	23	30
3	-	-	74	69
False Alarm	0.039	0.041	0.042	0.047
Hit rate	0.96	0.94	0.94	0.88
Missed detection	0.04	0.06	0.06	0.12
Specificity	0.96	0.96	0.96	0.95
Det. delay (samples)	11.45 ± 11.66	12.38 ± 10.78	15.13 ± 13.61	15.18 ± 14.09
Exec. time (s)	0.18 ± 0.005	0.72 ± 0.012	0.19 ± 0.004	0.72 ± 0.010

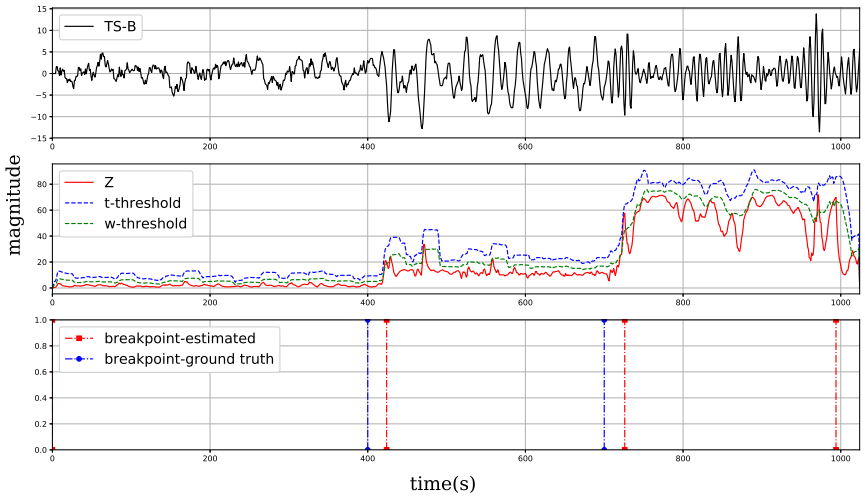


Figure 4.5: Breakpoints detection on TS-B

predicted change points versus the ground truth are presented. Figure 4.5 shows the detection on TS-B. From the previous section, it is known that TS-B has two change points. However, the figure shows that the proposed algorithm detects three change points, where the third detected point is considered as a false positive. This is acceptable since it can be seen by inspection from the first row of the graph that the time series looks actually non-stationary in term of the variance. The second row of the graphs shows the evolution of Z and the warning and trigger thresholds. In Figure 4.6, it can be seen that similar behavior, where few false alarms present, is also shown in the experiment on TS-C.

Furthermore, different behavior is observable in an experiment using TS-D. Figure 4.7 shows that the false alarm rate is somewhat higher than the other experiments using different time series. If the behavior of the time series is inspected closely, it certainly depicts a non-stationary behavior. This behavior is due to the fact that TS-D is a piece-wise stationary near-unit-root process with changing variance. Near-unit-root process means the process dynamic is close to unstable behavior. Therefore, it is expected that the SAFE approach considers some parts of the time-series as non-stationary, especially in the last part of the series. It is logical to consider that the process contains continuous non-stationarity, rather than just a change in the breakpoint. This way, the chosen online predictor can be continuously updated to adapt to the non-stationary process.

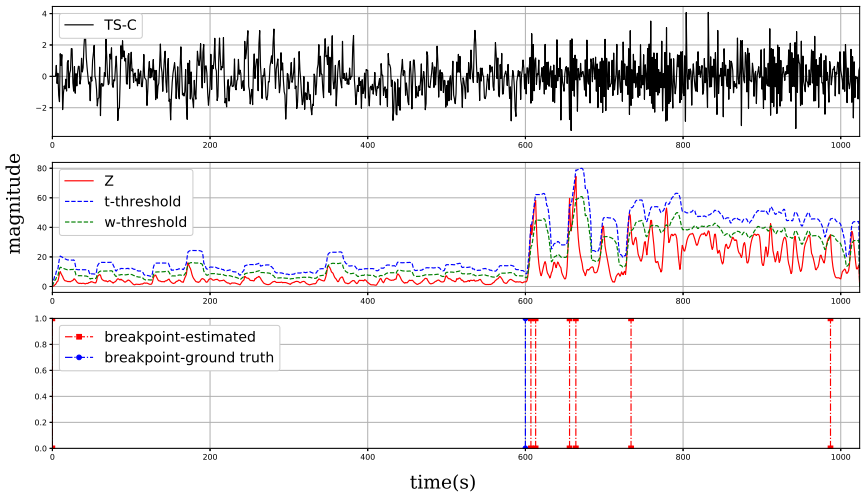


Figure 4.6: Breakpoints detection on TS-C

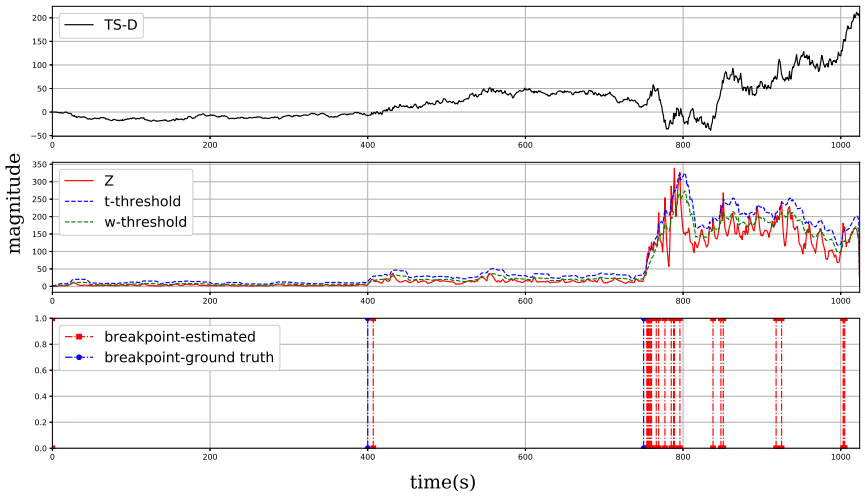


Figure 4.7: Breakpoints detection on TS-D

4.5.3 Choice of Predictors

The third setting of the experiments is employed to test which predictor gives better prediction performances. We use Linear-1, Linear-2, Nonlinear-1, and Nonlinear-2 datasets and test them with three different predictors namely Passive-Aggressive Regressors (PAR), Kernel Linear-SVR (KSVR), and Deep Neural Networks (DNN). Before the experiments are carried out, both parameters in Algorithm 5 and 6 have to be set. The parameters λ, T, W, β are set to be 0.3, 10, 20, and 0.1 respectively.

Furthermore, the parameters of the predictors also need initialization. For PAR, the aggressiveness parameter is set to 0.05. For KSVR, epsilon insensitive is used as the loss function, a L_2 regularizer with a constant equals to $1e^{-3}$. Finally, for the DNN, the number of hidden layers is set to 2, where each hidden layer contains 200 neurons, and use Rectified Linear Unit (ReLU) as the activation function. To avoid over-fitting, drop out regularizers with rate equals to 0.1 are implemented. All of the parameters were tuned using a similar off-line training scheme. The predictors are trained using the first 2000 samples and validated using the next 1000 samples. Since, in time series, it is not logically correct to train using data that come after the validation data, K-fold cross-validation is not suitable to implement. Basically, the predictors are trained until the validation errors stop decreasing. The trained predictors are then used as initial models that will be updated when they are triggered by the SAFE algorithm.

Finally, the rest of the data are used to test the predictors. The experimental results of these experiments are summarized in Table 4.5. The results show that the smallest prediction errors on all datasets are achieved using DNN as the predictor. However, the average execution time of this predictor is also shown to be the highest among all. In some applications, it might be crucial to have slow execution time. Accordingly, if we concern more about execution time than the prediction errors, then DNN might not be the best choice as a predictor. However, in an application such as traffic flow prediction, where the sampling time is 15 minutes, DNN is suitable as a predictor since it produces significantly lower prediction errors than the rest of the predictors do.

Figure 4.8 depicts the prediction performances on Linear-1 dataset. The first graph shows that all of the predictors are able to follow the ground truth closely. However, if we look at the errors on the second graph, the DNN error is fairly lower than those of other predictors although the off-line error of the DNN is higher than that of PAR. The next step is to compare the performance of the DNN with the baseline predictor, which is the predictor that is not updated. It can be seen in Figure 4.9 that while the error of the DNN stays constant, the error of the baseline predictor is drifting to a larger value. We can conclude that updating the SAFE approach is reliable for non-stationary time series

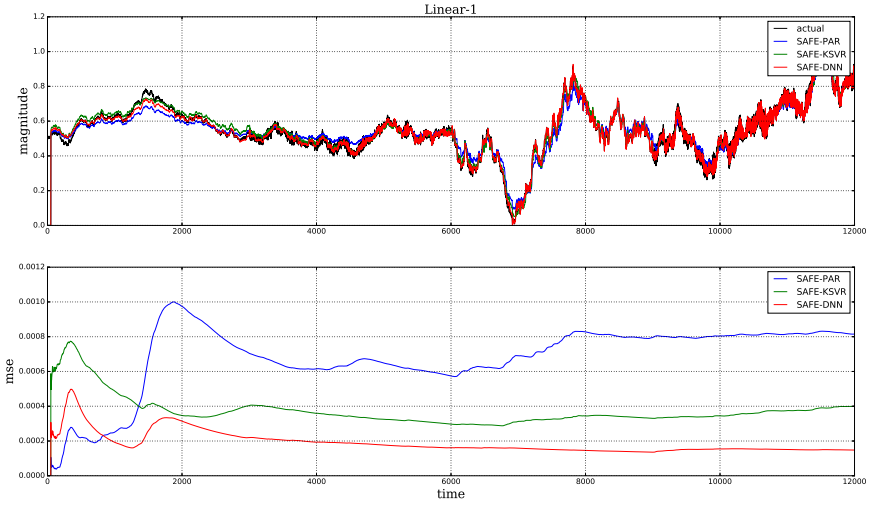


Figure 4.8: Prediction performance on Linear-1 dataset.

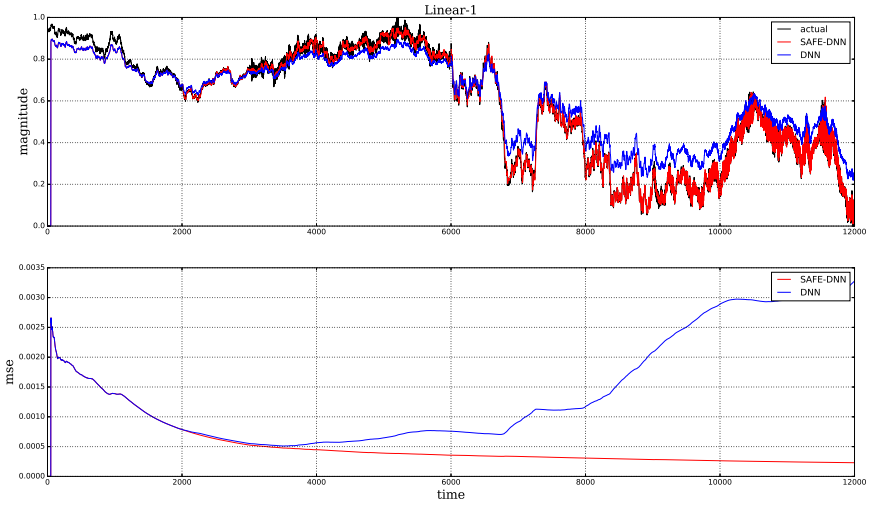


Figure 4.9: Comparison between SAFE-DNN and the baseline predictor on Linear-1 dataset.

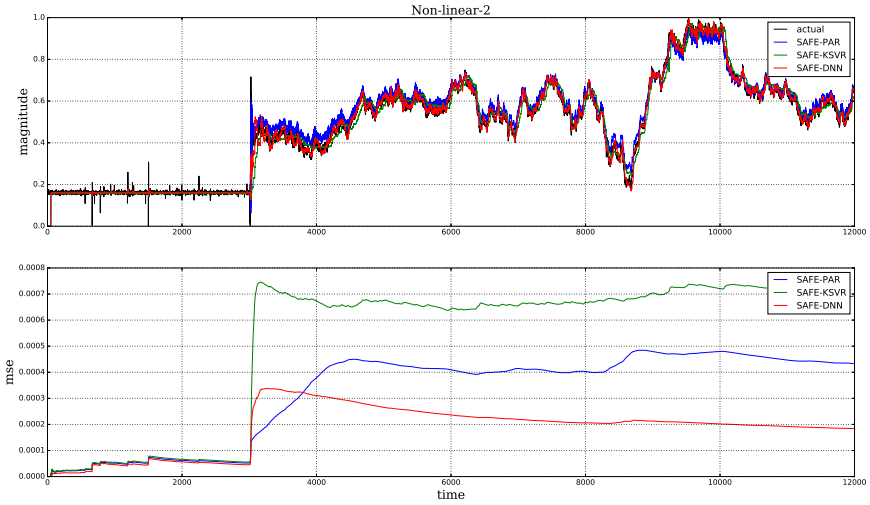


Figure 4.10: Prediction performance on Nonlinear-2 dataset.

prediction.

Lastly, Figure 4.10 and 4.11 shows that similar performances are shown in the nonlinear datasets. Although the predictor errors start around similar values, the DNN performs better in the long run. In addition, all of these prediction performances are achieved by updating the predictors when necessary only. This is reflected by the percentages of the update for all datasets that are less than 35%.

4.5.4 Real-world Data Experiments

The objective of the last set of experiments is to test the prediction performance of DNN on a real-world dataset under SAFE and time-domain FE. The dataset used is traffic flow dataset of San Francisco Bay Area, District 4, California. The data from January 1st 2011 to August 31st 2011 are used for training; September 1st 2011 to December 31st 2011 are used for validation; and January 1st 2012 to December 31st 2012 are used for testing.

The off-line predictors for both SAFE and time-domain FE in each dataset are identical. Next, the proposed approach is tested with traffic flow dataset. The baseline DNN is configured as follows: the number of hidden layers is 3; each hidden layer contains 125 neurons; and the drop-out rate equals to 0.5. The activation function of both the hidden

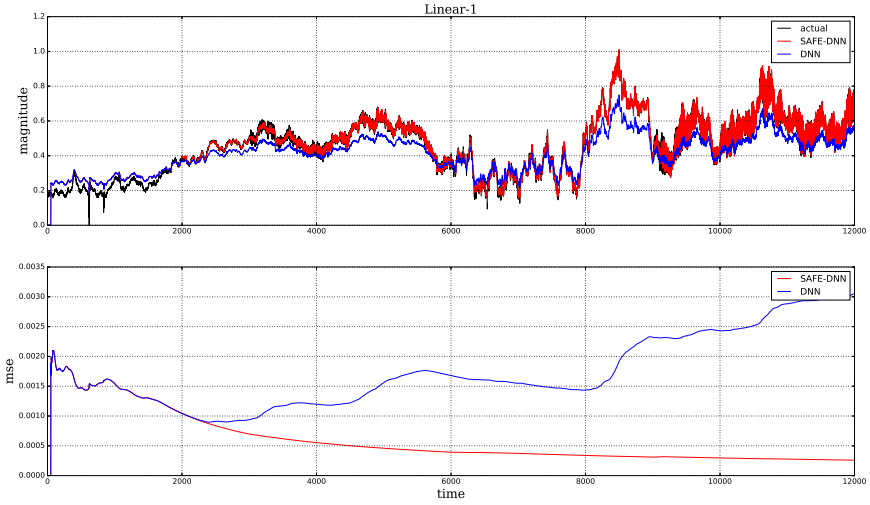


Figure 4.11: Comparison between SAFE-DNN and the baseline predictor Nonlinear-1 dataset.

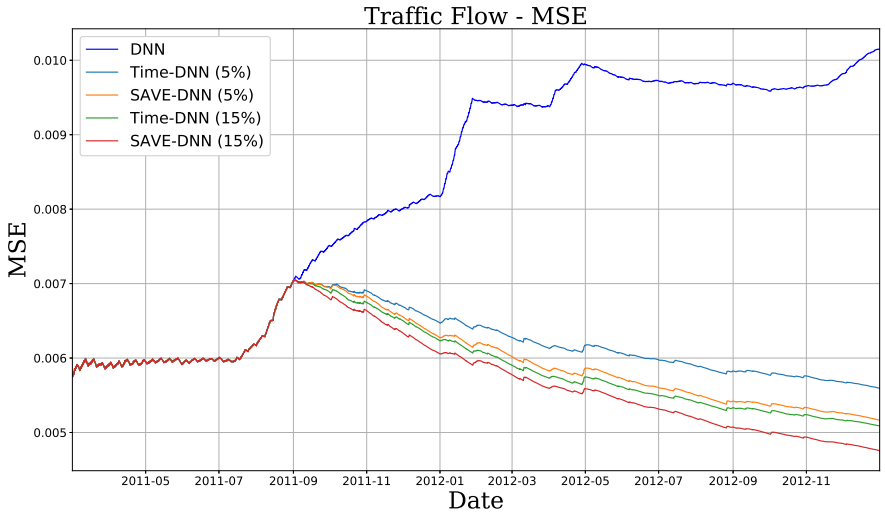


Figure 4.12: The evolution of errors on TF dataset.

Table 4.5: Summary of Non-stationary time-series prediction performance on the artificial datasets (Results over 20 simulations).

Model	SAFE-PAR		% update
	mse ($\times 10^{-3}$)	ex. time (s)	
Linear-1	3.70 ± 1.10	3.15 ± 0.22	17.72 ± 3.23
Linear-2	6.22 ± 1.73	3.44 ± 0.27	31.03 ± 4.07
Non-linear-1	4.12 ± 2.42	2.83 ± 0.20	20.39 ± 4.25
Non-linear-2	4.20 ± 0.92	2.64 ± 0.07	6.56 ± 1.68
Model	SAFE-KSVR		% update
	mse ($\times 10^{-3}$)	ex. time (s)	
Linear-1	2.40 ± 1.50	5.30 ± 0.92	17.72 ± 3.23
Linear-2	3.20 ± 1.42	9.25 ± 3.28	31.03 ± 4.07
Non-linear-1	1.76 ± 1.10	5.07 ± 1.38	20.39 ± 4.25
Non-linear-2	7.15 ± 9.90	3.55 ± 0.24	6.56 ± 1.68
Model	SAFE-DNN		% update
	mse ($\times 10^{-3}$)	ex. time (s)	
Linear-1	$\mathbf{1.45 \pm 1.10}$	35.65 ± 8.59	17.72 ± 3.23
Linear-2	$\mathbf{2.17 \pm 1.39}$	72.63 ± 24.70	31.03 ± 4.07
Non-linear-1	$\mathbf{0.90 \pm 0.91}$	42.33 ± 12.69	20.39 ± 4.25
Non-linear-2	$\mathbf{2.17 \pm 1.00}$	26.56 ± 5.56	6.56 ± 1.68

Table 4.6: Summary of Non-stationary time-series prediction performance on the real-world dataset.

Model	mse ($\times 10^{-3}$)	ex. time (s)	% update
DNN	4.06	-	-
	4.06	-	-
Time-DNN	2.24	482.93	5
	2.04	1518.53	15
SAFE-DNN	2.07	**434.85**	5
	1.84	1420.29	15

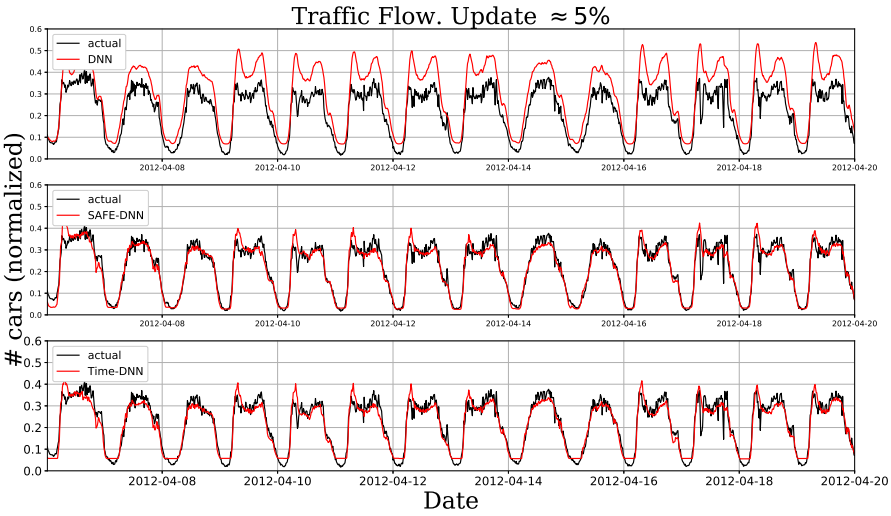


Figure 4.13: Traffic flow prediction with percentage of update ≈ 5%.

layers and the output layer is Rectified Linear Unit (ReLU) since it is known that the traffic flow values cannot be negative. Furthermore, five freeways are selected in these experiments. The results are shown in Table 4.6. In general, the combination of SAFE and DNN produces superior results in terms of errors and execution time. The evolution of the errors is shown in Figure 4.12. It can be seen that the errors start at the same level. However, the baseline error drifts away while the errors of the online predictors keep decreasing, and the SAFE-DNN error is always the lowest compared to the other errors.

Lastly, Figure 4.13 shows the prediction of traffic flow time-series using baseline predictor, SAFE approach, and time-domain FE approach. A portion of the prediction is selected to better illustrate the results. It can be seen that the baseline predictor does not produce acceptable traffic flow prediction while the SAFE-DNN and time-domain FE-DNN do. Although the predictions of both the approaches look similar, they are essentially different, especially in the valley parts of the traffic flow. It should be noted that this excellent prediction is obtained only by updating the predictor 5% throughout the experiments. This means we save around 95% of the processor or GPU cycles.

4.6 Conclusion

This chapter proposed an approach to actively detect non-stationarity for traffic flow prediction. The approach monitors the evolution of the spectral contents of the traffic flow time series using a distance function. Comprehensive experiments to validate our hypothesis and test the effectiveness of the proposed approach on artificial and real-world datasets have been successfully conducted.

The experiments show that the approach is able to achieve high long-term traffic flow prediction performances while significantly saving computational resources in terms of processor and GPU cycles. Although DNN requires more computational resources than other predictors do, it is clearly worth considering as an online predictor since its overall prediction errors are notably lower than those of the other predictors. The implementation of the proportional algorithm to variably include some data in the past makes the online adaptation of the predictors more flexible, i.e., there is no need to fix the batch size of the online training procedure. The proposed method can be used to work with any large-scale time-series, where distributed neural networks, e.g., DNN with multi-task learning, are appropriate.

Chapter 5

Multi-Step Traffic Flow Prediction[1]

5.1 Introduction

Accurate and timely traffic flow prediction is essential for traffic management and allows travelers to make better-informed travel decisions. In some applications, it is often necessary to predict traffic flow not only accurately but also several steps ahead in the future. For example, in order for the traffic patch to manage a congested road and develop contingency plans, traffic dispatch may need to estimate traffic conditions at least 30 minutes in advance. However, most traffic flow prediction approaches were developed with single-step prediction methods. The multi-step prediction problem is significantly more difficult than its single-step variant and is known to suffer from degradation in predictions the farther we go in future time-steps. Therefore, it is essential to develop multi-step prediction approaches to achieve accurate multi-step traffic flow prediction.

Multi-step time series prediction tasks are defined as tasks for predicting the next k values $[y_t, y_{t+1}, \ldots, y_{t+q}]$ given a historical time series $[y_t, y_{t-1}, \ldots, y_{t-p}]$, where $p, q > 1$ denote the past and future horizons. Generally, there are three main strategies for multi-step time series prediction: recursive, direct, and multi output [21]. In the recursive strategy, a one-step model is first trained to fit the following function:

[1]The contents of this chapter have been incorporated within a paper that has been submitted for publication. Arief Koesdwiady, Alaa El Khatib, Fakhri Karray, "Methods to Improve Multi-Step Time Series Prediction". Submitted to the International Joint Conference on Neural Networks (IJDNN) 2018. Submission date Feb. 1, 2018. The contents of this chapter have also been incorporated within a paper that has been uploaded to ArXiv and is going to be submitted for publication. Arief Koesdwiady, Fakhri Karray, "Improving Multi-Step Traffic Flow Prediction".

$$y_{t+1} = f(y_t, \ldots, y_{t-p}) \tag{5.1}$$

The learned model, f, predicts a multi-step time-series trajectory by repeatedly passing its predictions at one time step as input to the next time step. In the simple case where both the history and predictions are length-1 scalars, given $x(0)$, a model f predicts $\hat{x}(1)$ as $f(x(0))$, $\hat{x}(2)$ as $f(\hat{x}(1))$, and so on[2]. Due to accumulating errors and shifting input distribution, model predictions farther in the future increasingly drift from ground truth trajectories [151]. Moreover, there is a mismatch between what the model is optimized for, i.e., single-step error, and what it is used for, i.e., multi-step prediction, that gives rise to optimistic error estimates during training [139]. These weaknesses are present when the true single-step model is not identified during training [140], which is almost always the case in non-linear problems. Recent work showed that a learned model can be tuned to learn corrections to the drift patterns seen in the training data [151]. This is an iterative training process, in which the training set is repeatedly augmented with additional data points of the form $(\hat{x}(t), x(t+1))$, where $\hat{x}(t)$ represents predictions of the model when applied to the training set. When the model is applied recursively to the training set points, it generates prediction trajectories of some length. The intuition of this iterative training process is that by augmenting the training set with samples of these predicted trajectories, coupled with the next-step ground truth values, the model can learn to correct the drift patterns in its predicted trajectories.

Alternatively, one can do without the recursive process by learning a separate model to directly predict each time-step in the future, i.e., the direct strategy, which is given as

$$y_{t+h} = f_h(y_t, \ldots, y_{t-p}) \tag{5.2}$$

where $h \in 1, \ldots, H$ is the predictions horizon. In this strategy, multi-step predictions are obtained by concatenating the H predictions. Unlike the recursive strategy, the direct strategy does not suffer from accumulating errors since it does not use any predicted values for the subsequent predictions. However, there are two major weaknesses possessed by this strategy. First, since each model is learned independently, dependencies between two distant horizons are not modeled. Second, this strategy requires a large computational resources, i.e., time and space, since the number of models depends on the size of the prediction horizon.

[2]This simple case in presenting recursive prediction is used in this work, but the ideas discussed apply to the general case.

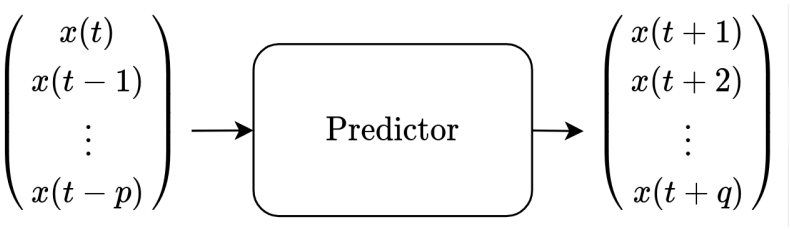


Figure 5.1: Multi-output strategy.

The third strategy is the multi-output strategy. This strategy is defined as the problem of finding a model f that predicts the future $[x(t+1), x(t+2), \cdots, x(t+q)]^\top$ given the historical data $[x(t), x(t-1), \cdots, x(t-p)]^\top$. The strategy requires a model that is able to produce multi-step predictions simultaneously, as depicted in Figure 5.1. This way, each prediction uses the actual observations rather than the predicted ones. Therefore, accumulated errors are not of concern in this strategy. Moreover, this strategy can learn the dependency between inputs and outputs as well as among outputs. Hence, the strategy involves more complex models than the recursive one does, which directly translates to a slower training process and requires more training data to avoid over-fitting. While in some cases the direct and multi-output strategies can avoid some of the pitfalls of the recursive strategy, these models still suffer from degrading performance in the farther time-steps. Intuitively, there is a higher uncertainty associated with the farther future that makes it more difficult to forecast. Moreover, direct models can suffer from higher variance [103]. Researchers have attempted to analyze theoretically and empirically the differences between recursive and direct/multi-output approaches and understand which would be more appropriate for a given problem [141], but the results of this effort so far have been inconclusive. In practice, all approaches continue to suffer from increasingly drifting predictions for farther time-steps.

Recently, an approach to counter the drifts in trajectories of multi-step predictions called Data as Demonstrator (DaD) is proposed in [151], specifically in the context of recursive prediction models. The underlying idea in their approach is to use the drift patterns seen when a trained model is applied to the training data to tune that model such that it can compensate for these drifts. Another way to look at it is as a data augmentation technique that alleviates the mismatch between training and testing distributions. Inspired by this, two approaches to enhance multi-step prediction accuracy are introduced in this thesis. In the context of recursive models, a time-step-augmented model that implicitly learns to associate a different corrective action with different future time-steps is proposed. The model is an extension of the approach proposed in [151]. This is also related to the

Rectify method proposed in [142], where a direct model is trained to correct the predictions of a recursive model at each time-step in the prediction trajectory. In the second approach, a data augmentation method that enhances multi-step prediction accuracy in multi-output models is proposed. Here, a conditional generative adversarial network (C-GAN) is used to learn a generator model that can mimic the historical patterns corresponding to the future patterns seen in the training data. Subsequently, this model is used to generate new history-future pairs that are aggregated with the original training data.

The main contributions in this work are summarized as follows:

- An extension to the algorithm presented in [151], where information about the current time-step prediction is augmented in the model. This extension is called Conditional-DaD (C-DaD).
- A novel approach for generating new history-future pairs of data that are aggregated with the original training data using C-GAN.
- Comprehensive traffic flow prediction experiments involving recursive, direct, and multi-output strategies. In the recursive strategy, the vanilla approach, DaD, and C-DaD are experimented. Furthermore, the vanilla direct strategy and its modification using recursive strategy, namely *Hybrid* method, are also presented. Finally, the proposed method C-GAN is compared with noise-augmented training strategy as well as the vanilla multi-output strategy.

The rest of this chapter is structured as follows. Section 5.2 introduces the two proposed approaches. Section 5.3 describes the experimental setup and the data sets used to evaluate the proposed models. In Section 5.4, the experimental results are presented and discussed. Finally, the chapter is concluded with some observations in Section 5.5.

5.2 Methodology

In this section, two methods to improve multi-step time-series prediction are introduced. An approach to improving recursive multi-step prediction, called Conditional-DaD (C-DaD), followed by a conditional-GAN-based data augmentation approach to improving multi-output multi-step prediction are introduced.

5.2.1 Conditional-DaD

One weakness of the approach presented in [151], and recursive prediction generally, is that it does not take into consideration the number of steps predicted by the model so far. The amount of correction the model needs to add differs from time-step to another along a multi-step prediction trajectory, since the deviation from ground truth is less acute in early steps. Therefore, the model stands to benefit from having information about the current time-step along the prediction trajectory. In particular, the amount of correction the model needs to add is affected by the number of time-steps that have passed in which the model used its prediction as input to the next time-step.

Based on this, an extension to DaD called C-DaD, in which the input is augmented with a representation of the current time-step, is proposed. For length-1, scalar history, $x(t)$, the single-step prediction model is modified to accept an augmented vector, $[x^n(t), v_n]^\top$, where n is the prediction time-step, and v_n is a representation of n. In the presentation and experiments, $v_n = n$ is used. An illustration of this is shown in Fig. 5.2.

A C-DaD model learns a single mapping that is function of the number of predicted values that have been recycled as input (and also function of the time-series history). This arrangement allows the model to output different $x(t+1)$ for the same $x(t)$ input, depending on the current time-step along the prediction trajectory, and, hence, allows the model to learn different corrections for different time-steps. This setup differs from the parameterized recursive prediction approach, where a different set of parameters are learned for each time-step in the future.

Training a C-DaD model follows a similar process to the meta algorithm proposed in [151], the difference being in the addition of the time-step representation. Algorithm 7 describes this process. In short, the time-step-augmented training data is generated by forward-passing the original training data through a base model. Next, a C-DaD model is iteratively trained, and a new augmented training data set is generated every epoch by passing the original data through the previous C-DaD model. Furthermore, the final model is selected based on the performances of all models on the validation data set.

5.2.2 Conditional-GAN Data Augmentation

In some applications, recursive models do not perform well compared to other approaches such as multi-output models [103]. Nonetheless, the performance of multi-output models suffers degradation as the prediction time-step increases. In this section, a C-GAN-based

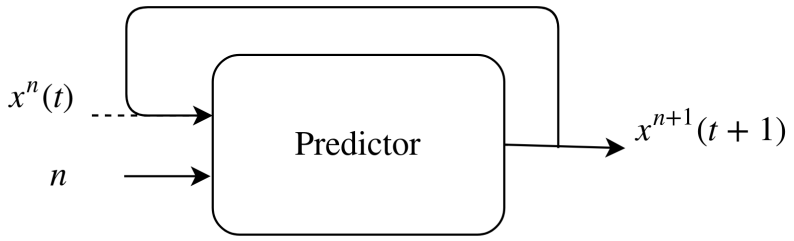


Figure 5.2: C-DaD recursive prediction model.

Algorithm 7 C-DaD training process.

Input:

- X—time-series points, $x(t)$.
- *predict_recursively*—procedure that takes as input a sequence of points, X, an integer, N, and a single-step prediction model, *net*, and outputs the length-N recursive predictions of *net* applied to the data points in X. The output of this is of the form $\hat{x}^n(t)$, where n indicates the recursion index.
- *predict_recursively_aug*—similar procedure to *predict_recursively*, except that the model is applied to time-step-augmented inputs X_{aug} of the form $[x^n(t), n]^\top$.

Initialize:

- N—number of recursive time steps to use.
- epochs—number of iterations to train the C-DaD model.

1: Use X to build a training set D of the form $(x(t), x(t+1))$.
2: Initialize single-step-prediction base model, M_{base}, and train it using D.
3: $\hat{X}^{(N)} \leftarrow$ *predict_recursively*(X, N, M_{base})
4: Use X and $\hat{X}^{(N)}$ to build a training set D_{aug} of the form $([\hat{x}^n(t), n]^\top, x(t+1))$, $n \in [0, N]$
5: Initialize single-step-prediction augmented model, M_0.
6: **for** $n = 1, \ldots, K$ **do**
7: Build a time-step-augmented inputs X_{aug}
8: $\hat{X}^{(N)} \leftarrow predict\_recursively\_aug(X_{aug}, N, M_{n-1})$
9: Use X_{aug} and $\hat{X}^{(N)}$ to build a training set D_{aug}
10: $M_n = train(D_{aug})$
11: **end for**
12: **return** M_n with lowest error on the validation data.

data augmentation approach to improve multi-output multi-step time series prediction is introduced.

The multi-output strategy requires a model that is able to produce multi-step predictions simultaneously. This way, each prediction uses the actual observations rather than the predicted ones. Therefore, accumulated errors are not of concern in this strategy. Moreover, this strategy can learn the dependency between inputs and outputs as well as among outputs. Hence, the strategy involves more complex models than the recursive one does, which directly translates to a slower training process and requires more training data to avoid over-fitting.

Similar to the recursive strategy, a data augmentation method can be applied to improve the multi-step time series prediction. One simple way to augment the data is to contaminate the features with noise, and pair them with the actual labels. This method can increase the multi-step prediction performance if the noise is carefully chosen. Poor choice of noise, however, may significantly degrade the prediction performance. In this work, an alternative method, i.e., Generative Adversarial Network (GAN) [55], to augment the data by learning a distribution over input conditioned on the output is proposed.

GAN is a framework for estimating a distribution in an adversarial manner. It simultaneously trains two models, namely a generative model G and discriminative model D. The discriminative model is trained to maximize the probability of assigning appropriate labels for both samples coming from the training data and generative model. Simultaneously, the generative model is trained to minimize $\log(1 - D(G(z))$, where z is a random sample from an input noise distribution. Both D and G are playing a two-player minimax game with the value function $V(G, D)$ given as follows:

$$\begin{aligned}\min_G \max_D V(D,G) =& \mathbb{E}_{x\sim p_{data}(x)}\left[\log D(x)\right] \\ &+ \mathbb{E}_{z\sim p_z(z)}\left[\log(1 - D(G(z)))\right] \end{aligned} \tag{5.3}$$

An extension of GAN, namely conditional generative adversarial nets (C-GAN), is proposed in [107]. In this extension, both G and D are conditioned on some extra information, which can be any kind of auxiliary information such as class labels. There have been some research applying C-GAN to discrete labels [39], text [126], and images [162]. In this work,C-GAN is trained to generate inputs (historical data) given the actual labels (future data). In both the generative and discriminative models, the actual labels are concatenated with noise. This idea is illustrated in Figure 5.3, where $x(t + 1)$ is the label (future data), $\hat{x}(t)$ is the generated inputs (past data). The pair of generated inputs and actual labels then are augmented in the original training data for multi-output time series prediction.

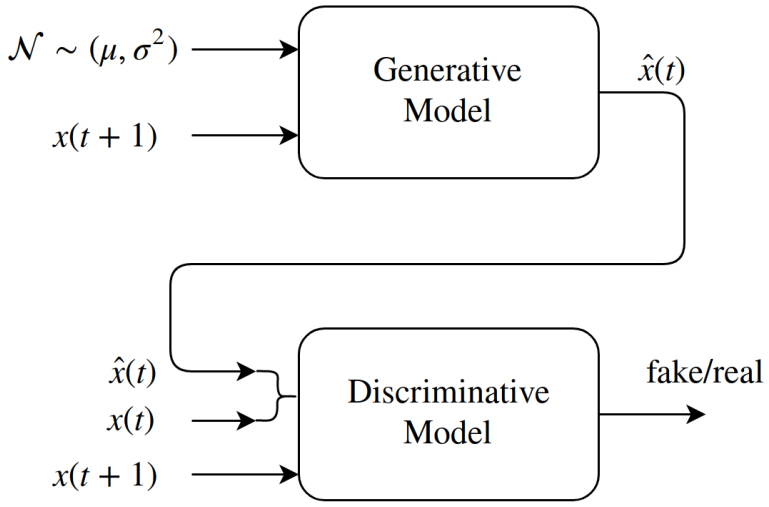


Figure 5.3: C-GAN for generating time series data.

Using this method, an infinite amount of data to enhance the predictor performance can be generated. In addition, using the generated data, there is no need any special treatments in the training process. It can be done in a standard multi-output training without any iterative training processes.

5.3 Datasets and Experimental Settings

To test the performance of the proposed methods, we conduct experiments using a traffic flow data set:

- **Traffic Flow**: the data set was downloaded from the Caltrans Performance Measurements Systems (PeMS) [1]. The original traffic flow was sampled every 30 seconds. These data were aggregated into 5-min duration by PeMS. Highway Capacity Manual 2010 [102] recommends to aggregate the data further into 15-min duration. We collected the traffic flow data of a freeway from January 1st 2011 to December 31st 2012. We use data from from January 1st 2011-from August 31st 2011 for training, September 1st 2011-December 31st 2011 for validation, and the rest for testing.

Three sets of experiments are conducted using the data set. In the first set, the vanilla recursive strategy, DaD, and C-DaD are implemented on the data set. The main goal

of these experiments is to investigate which strategy performs better in the recursive setting. Next, the vanilla direct strategy and Hybrid approach are applied to the data set. Subsequently, the proposed C-GAN data augmentation is applied and compared with the vanilla multi-output strategy and noise data augmentation model. The number of the time steps for the multi-step prediction is chosen to be 8. Furthermore, the performances of the best models from of each strategy are compared and analyzed. The performance of each of the experiments is evaluated using Mean Squared Error (MSE) and Mean Absolute Error (MAE). To illustrate the superiority of the proposed methods, the percentages of improvement of the errors with respect to the baselines are computed.

5.4 Results and Analysis

In the first set of the experiments, three recursive approaches are applied to the data set. Each approach uses similar base predictor, which is a deep neural network (DNN). To have fair comparisons, the configurations of the DNN for all of the approaches, in terms of number of hidden layers, number of hidden units, activation function, and tricks used for the training, are kept identical. The main difference is that in the C-DaD approach the input is augmented with the time-step information, which means there are extra weights associated with this input.

The base DNN is configured to have 2 hidden layers where each layer contains 150 hidden units, and the activation function is selected to be ReLU. Since it is a regression problem, a linear activation is used in the output layer with MSE as the loss function. Furthermore, the data are min-max normalized between 0 and 1. Moreover, to avoid the model from over-fitting too easily, drop-out regularizers with rate equals to 0.1 are implemented on each layer. In addition, the Adam [79] algorithm is used for the gradient-based optimization.

The summary of the performance of the recursive approaches can be seen in Table 5.1. The table shows that, with respect to the vanilla recursive approach, the DaD and C-DaD approaches have successfully improved the overall MSE and MAE. This shows that reusing the prediction results as the input data, together with the original training data, leads to improvement in the performances. Furthermore, augmenting the information of the time step in the model, i.e., C-DaD approach, can further improve the performances for more than 2 times in the MSE and almost 1.5 times in the MAE, as can be seen in the table. These performances are achieved after 25 and 29 iterations in the C-DaD and DaD approaches, respectively.

Table 5.1: Summary of recursive multi-step prediction performances.

Perf. / Models	MSE	% Improv.	MAE	% Improv.
Recursive	0.0101	-	0.0781	-
DaD	0.0092	8.16	0.0627	19.64
C-DaD	0.0078	22.92	0.0563	27.89

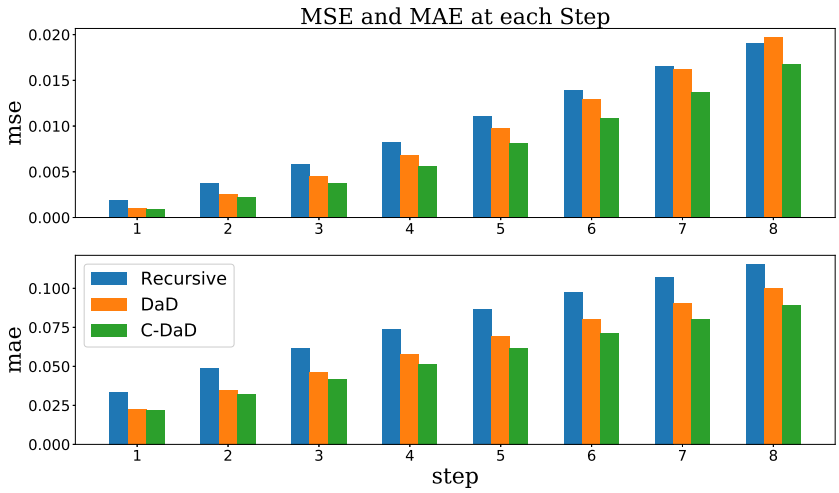


Figure 5.4: MSE (top) and MAE (bottom) at each step for the recursive approaches.

Figure 5.4 depicts the error occurs at each time step. It shows that, in the early step, both the DaD and C-DaD approaches perform significantly better than the vanilla recursive does. However, at time step equals to 8, the MSE of the DaD approach is worse than that of the vanilla recursive, while the C-DaD approach is able to maintain its performances all the way through the last step. This is not the case for the MAE, where both the DaD and C-DaD approaches are able to maintain its performances at all time steps.

The results of the traffic flow predictions for the recursive approaches can be seen in Figure 5.5. This figure presents traffic flow predictions at time step 1 and 8. At time step 1, all the approaches produce similar predictions, which are very close to the actual traffic flow. However, at time 8, only the C-DaD approach is showing an acceptable prediction. Indeed, the augmentation of the time-step information provides an extra dimension to the model, which helps the model to understand the state of the prediction and learn better.

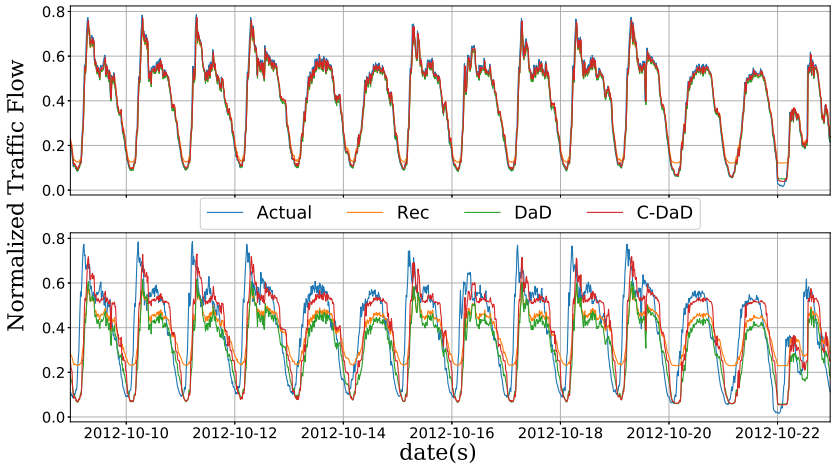


Figure 5.5: Recursive approaches traffic flow prediction results at time step 1 (top) and 8 (bottom).

Table 5.2: Summary of direct multi-step prediction performances.

Perf. / Models	MSE	% Improv.	MAE	% Improv.
Direct	0.0090	-	0.0715	-
Hybrid	0.0082	8.68	0.0674	5.63

It can be concluded that adding this extra dimension is worth the effort.

In the second set of experiments, two direct approaches, namely vanilla direct and Hybrid approaches, are tested. Similar base predictors as the previous set of experiments are used. The number of models trained in this approach depends on the size of the future horizon. Since the number of time steps is 8, then the number of models in each approach will be 8. The main difference between the vanilla direct and Hybrid approaches is in the size of the input. The number of input in the Hybrid increases as the time step increases while the number of input in the vanilla direct is static.

The performance of the direct approaches is summarized in Table 5.2. In this table, it can be seen that the vanilla direct approach has smaller errors than the vanilla recursive does. This is expected since in the vanilla direct approach the accumulating error problem does not exists. Each time-step prediction is handled independently by each model.

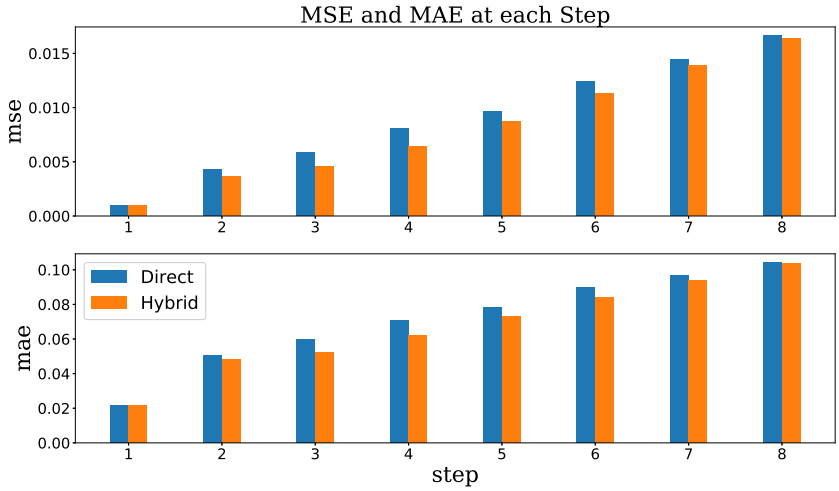


Figure 5.6: MSE (top) and MAE (bottom) at each step for the direct approaches.

Furthermore, the Hybrid approach improves the vanilla direct performance considerably. However, if it is compared with the best performance in the recursive approach, C-DaD is still shown its superiority. This may be attributed to the accumulating errors when the previous predictions are used as inputs in the subsequent models.

From Figure 5.6, it can be seen that the errors in the first step are identical. This is possible because the models in this step are practically the same since the previous predictions are not utilized yet. Overall, the Hybrid approach improves the prediction errors in all time steps. However, a closer look suggests that the Hybrid model performances degrade as the time step increases. Indeed, at the later steps the accumulating errors dominate the input of the model. In this type of approach, the performance in the next step highly depends on the one in the previous model.

Figure 5.7 shows that, initially, the Hybrid approach produces acceptable prediction, which is not the case in the last step. This is reciprocal with the errors shown in Figure 5.6, where there is almost no improvement in MSE and MAE achieved in the last step. In comparison with C-DaD, this method is computationally inefficient since it requires several models for multi-step predictions. With 8 times less computational efforts, the C-DaD approaches performs considerably well than the Hybrid method does.

The last set of experiments involves three multi-output approaches: vanilla multi, noise-augmented, and C-GAN approaches. The base DNN is configured to have 2 hidden layers where each layer contains 150 hidden units, and the activation function is selected to be

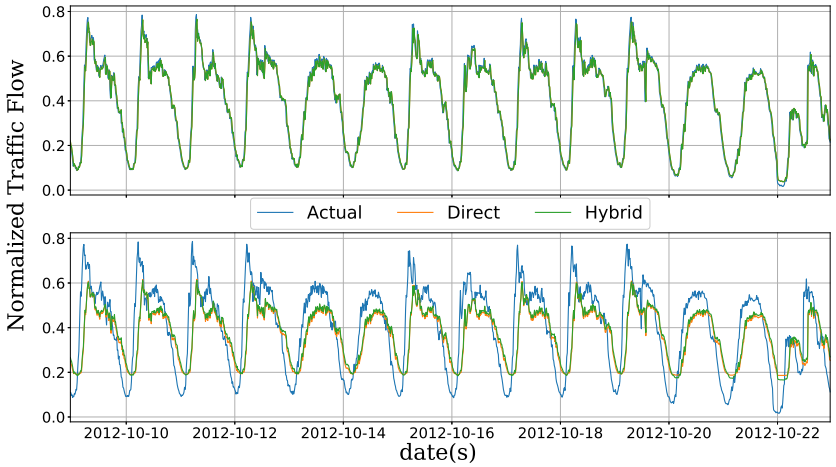


Figure 5.7: Direct approaches traffic flow prediction results at time step 1 (top) and 8 (bottom).

ReLU. The three approaches use identical models, including the output layer size. In the noise-augmented approach, the data are contaminated with a Gaussian noise with mean equals to 0 and variance equals to 0.1. After several trials, this variance is found to produce the best performance on the validation data set. Meanwhile, the discriminative and generative models of the C-GAN use DNN with similar configuration as the base DNN. The important aspect of training the C-GAN is the learning rates of both the discriminative and generative models. Usually, the discriminative model is configured to learn faster than the generative model. This way the discriminative loss stays low, which makes it stays ahead of discriminating new strange representations from the generative model. The evolution of the losses in the DC-GAN is depicted in Figure 5.8. It can be seen that the losses of both the discriminative and the generative models converge. Furthermore, the C-GAN accuracy converges to 50%, which means the discriminator is not able to distinguish the data generated by the generative model from the actual data. Therefore, it can be concluded that the generative model acts as a distribution that mimics the training data.

The overall performances of the multi-output traffic flow prediction approaches are summarized in Table 5.3. So far, the lowest MSE using the vanilla approaches is obtained by the multi-output approach. It is attributed to the fact that in the multi-output setting, the accumulating errors problem does not exist and the dependencies between time steps are modeled. In the noise-augmented approach, a poor choice of noise may significantly degrade

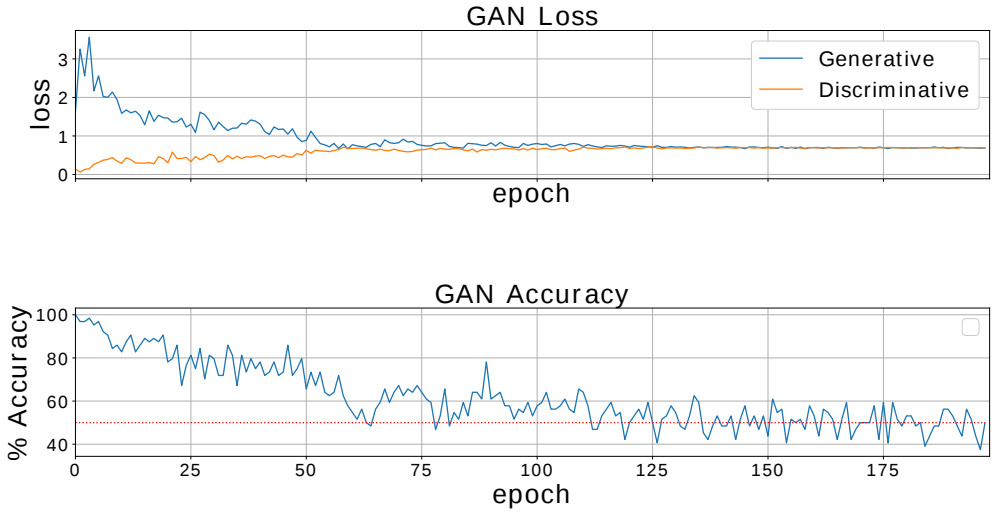


Figure 5.8: GAN loss and accuracy progressions.

Table 5.3: Summary of multi-output multi-step prediction performances.

Perf. / Models	MSE	% Improv.	MAE	% Improv.
Multi	0.0089	-	0.0718	-
Noise	0.0082	8.13	0.0671	6.57
GAN	0.0072	18.47	0.0576	19.71

the prediction performances. However, in this experiment, the noise has been successfully chosen as it is evident in the prediction performances improvements. Furthermore, the best improvement is achieved when the original data is augmented with the one generated by the generative model. Therefore, C-GAN can be seen as an intelligent way for data augmentation.

Figure 5.9 depicts the MSE and MAE of the multi-output approaches at all time steps. Both the noise-augmented and C-GAN approaches are consistently produce improved traffic flow predictions all the way through the all time steps. Furthermore, in Figure 5.10, it can be seen that the performances of the vanilla multi and noise-augmented approaches are poor in the first time step. Indeed, learning several time-steps simultaneously is more difficult than learning 1 step only as it is done in the recursive and direct approaches. However, the proposed C-GAN approach is able to significantly improve the early time

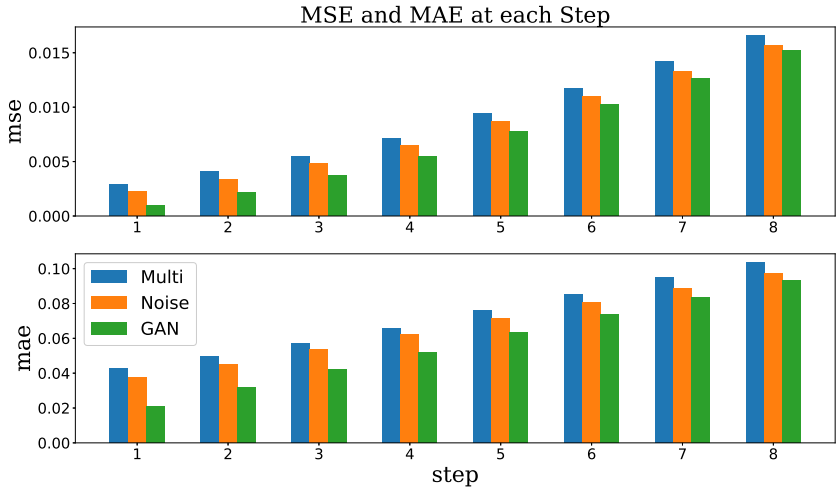


Figure 5.9: MSE (top) and MAE (bottom) at each step for the multi approaches.

step predictions and overall performances.

Finally, the comparison of C-DaD, Hybrid, and C-GAN approaches is depicted in Figure 5.11. This figure shows that the C-GAN approach has shown its superiority in term of MSE at all time steps. However, in term of MAE, the C-DaD approach is better at the later steps compared to the C-GAN approach. Indeed, based on the prediction plots in Figure 5.5 and Figure 5.10, it can be seen that the C-DaD approach produce better traffic flow prediction than the C-GAN does. In [158], it is suggested that MAE is more natural and quite often MSE can be misleading and is not a good indicator of average model performance because it is a function of two characteristics of a set of errors, rather than of one. In addition, [29] demonstrates that the MSE is more appropriate to represent model performance when the error distribution is expected to be Gaussian.

5.5 Conclusions

This chapter proposed two methods to improve multi-step traffic flow predictions: C-DaD and C-GAN approaches. The first approach is developed using recursive strategy and inspired by previous work [151]. This approach augments the information about the current time step and follows a similar training process to the meta-algorithm proposed in [151]. The second model is developed using multi-output strategy and utilizes the ability

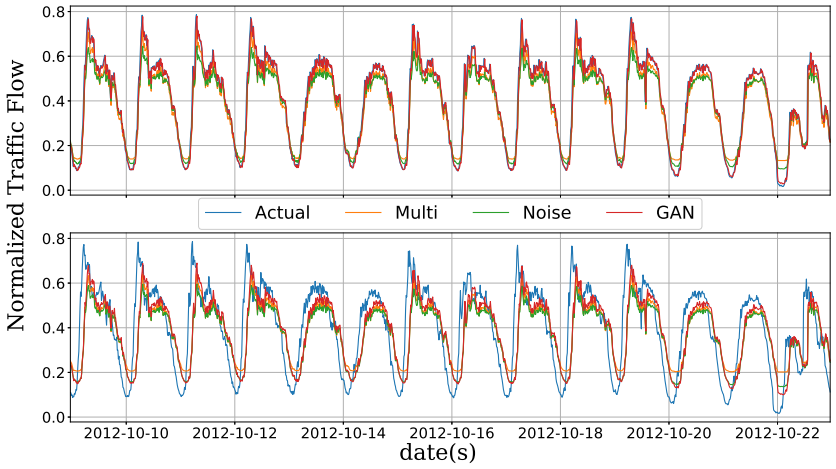


Figure 5.10: Multi approaches traffic flow prediction results at time step 1 (top) and 8 (bottom).

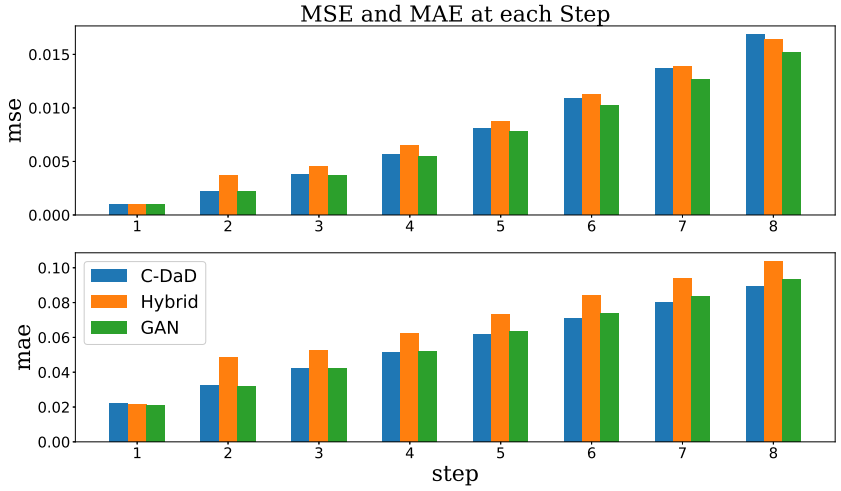


Figure 5.11: MSE (top) and MAE (bottom) at each step for the C-DaD, Hybrid, and C-GAN approaches.

of GAN in mimicking a data set distribution. The C-GAN model is developed to generate historical data conditioned on the future data. This way, the original data set can be enriched with an infinite amount of historical-future pairs of data for training purposes.

The experiments show that the proposed approaches are able to improve multi-step traffic predictions relative to their vanilla approaches. Moreover, in term of MSE the C-GAN approach performs better than the all of the approaches. However, in the latter steps, the MAE of the C-DaD is lower than all the experimented approaches. Compared to the C-DaD, the training of the C-GAN approach is fairly simpler since it does not require iterative training once the new data are generated. However, there are applications where it is more efficient to use recursive model, such as for video sequence prediction, and in such application recursive prediction can benefit from the improvement offered by the C-DaD approach.

Chapter 6

Conclusions and Future Directions

Smart mobility arises as a solution to overcome transportation problem in cities. Intelligent Transportation System (ITS) plays an important role in realizing smart mobility in smart city. Moreover, the development of smart mobility along with the Internet of Car (IoC) realization opens several doors for researchers to explore the implementation of Artificial Intelligence (AI) technology in transportation systems. However, this development is faced with critical challenges, namely big data. The hype and hope of big data is a transformation in the knowledge and governance of cities through the creation of a data deluge that seeks to provide much more sophisticated, wider-scale, finer-grained, and real-time understanding of city traffic. Nevertheless, there has not been any well-defined and concrete approach to take advantage of the big data phenomenon to assist the travelers and authorities in the decision-making process. In fact, the main bottleneck on this path is the fundamental challenges of traffic prediction methods that need thorough consideration.

The main objective of the research work in this thesis is to develop a traffic flow prediction framework under big data environment. The proposed framework was composed of four main modules that tackle major data-driven traffic flow prediction problems: the fusion of hard data for traffic flow prediction; the fusion of soft data for traffic flow prediction; non-stationary traffic flow prediction; and multi-step traffic flow prediction. The first module is developed to tackle the inherent big data problems and dealing with the uncertainty in the predictions. It relies on the ability of deep learning approaches in handling huge amounts of data generated by a large-scale and complex transportation system with limited prior knowledge. The second module fuses the streams of data and event-based data using Dempster-Shaffer Evidence Theory (DSET). One of the main features of the DSET is its ability to capture the uncertainty in the probability. Furthermore, the third module consists of a method to detect nonstationarities in the traffic flow and an algo-

rithm to perform online adaptations of the prediction model. Finally, the last module is developed to improve multi-step traffic flow prediction in the recursive and multi-output settings.

To summarize, we present novel approaches to handle large-scale traffic flow prediction using hard and soft data. We propose an extension to the classical Dempster's Rule of Combination to tackle the reliability issue of the information coming from soft data. To tackle the non-stationarity aspect of traffic flow data, we develop a novel approach to detect nonstationarity that is coupled with online learning methods. The developed method is able to improve traffic flow prediction performances while mitigating the well-known Catastrophic Forgetting problem. Furthermore, we propose two new approaches to predicting traffic flow multi-step ahead in the future. One of the approach, i.e., C-GAN approach, is the first attempt on mimicking a distribution of a time series given its labels using generative adversarial networks. The proposed approaches are scalable and work for other time series prediction problems. These whole proposed approaches have not been done for large-scale traffic flow prediction problem in particular, and time series prediction problem in general.

The performances of the proposed approaches were evaluated by running them on various experimental scenarios. The results showed that the proposed methods could successfully improve the quality of traffic flow predictions in real-world situations. The rest of this chapter highlights the contributions made in this thesis, discuss the main contributions, and determines the future directions.

6.1 Contributions Highlights

The main contributions of this thesis are summarized as follows:

- **Large-Scale Traffic Flow Prediction using Hard Data**: a modular big-data-based traffic flow prediction architecture is proposed. A Deep Belief Network is used as the main approach underpinning the traffic flow prediction process. The DBN has shown its capability in handling huge amounts of data generated by a large-scale and complex transportation system with limited prior knowledge. Furthermore, the proposed decision-level data fusion scheme applied to the traffic flow and weather data is able to improve the quality of the traffic flow prediction.
- **Large-Scale Traffic Flow Prediction using Soft Data**: an approach capable of acquiring the various data and information, processing the data for prediction,

and combining multiple evidences to enhance traffic flow prediction is proposed. The applicability of the proposed approach is tested using real data obtained from authorities and social media. Moreover, the proposed approach is able to handle transportation big data problem as well as unreliable data introduced by soft data using Dempster-Shafer Evidence Theory (DSET).

- **Non-stationary Traffic Flow Prediction**: a method to detect non-stationarities in the traffic flow based on the evolution of the spectral contents of the traffic flow time series is introduced. Subsequently, an online learning framework that efficiently combines the detection method with online machine learning methods is proposed. Furthermore, an algorithm to proportionally include some past data in the adaption process to overcome the *Catastrophic Forgetting* problem is also proposed.

- **Multi-step Traffic Flow Prediction**: approaches to improve multi-step traffic flow prediction is proposed. In particular, an extension to the algorithm presented in [151], where information about the current time-step prediction is augmented in the model is developed. This extension is called Conditional-DaD (C-DaD). Furthermore, a novel approach of generating new history-future pairs of data that are aggregated with the original traffic flow training data using Conditional-GAN (C-GAN) is proposed.

6.2 Future Research Directions

In the following, several interesting directions for future work are presented:

- The proposed deep-learning model could be extended to include more data such as traffic speed. This way, the traffic density can be derived and used to determine the traffic state of the road. Once the traffic state is known, time-space diagrams can be created to give travel time, delay, and queue lengths of a road.

- In this work, the categorical data, e.g., weather conditions, traffic events, were encoded using integer encoding. This encoding assumes natural ordered relationship between each other. However, this might not be the correct approach and prevent the performance from achieving higher results. One of the alternative ways is to use one-hot encoding.

- Traffic anomaly is defined as deviations of traffic from normal traffic patters. The results in this work can also be extended for detecting traffic flow anomalies. This is

the first step to proactively assess the imminent occurrence of traffic incidents. Furthermore, traffic flow anomaly detection can also be used to monitor the functionality of traffic sensors for predictive maintenance.

- The traffic flow prediction results produced by the proposed framework can be utilized to assist authorities in the decision making. Specifically, the accurate and reliable prediction results might be used to control the traffic flow by means of data-driven traffic control systems.

References

[1] Caltrans Performance Measurement Systems (PeMS). Available from *http://pems.dot.ca.gov/*. Accessed January 06, 2016.

[2] Federal Highway Administration. How Do Weather Events Impact Roads?. Available from *http://www.ops.fhwa.dot.gov/weather/q1_roadimpact.htm*. Accessed June 20, 2015 .

[3] IBM Infrastructure and Citizen Services. Available from *https://www.ibm.com/industries/government/infrastructure-citizen-services/transportation-management*. Accessed on November 5, 2017 .

[4] The Mesowest Dataset. Available from *http://mesowest.utah.edu/*. Accessed June 15, 2015.

[5] Transport Outlook 2012: Seamless Transport for Greener Growth. Organisation for Economic Co-operation and Development. Accessed June 20, 2015 .

[6] UN State of World Cities report. Available from *http://www.unhabitat.org/pmss/listItemDetails.aspx?publicationID=3387*. Accessed June 20, 2015 .

[7] Sudeshna Adak. Time-dependent spectral analysis of nonstationary time series. *Journal of the American Statistical Association*, 93(444):1488–1501, 1998.

[8] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831, 2015.

[9] Cesare Alippi, Giacomo Boracchi, and Manuel Roveri. A just-in-time adaptive classification system based on the intersection of confidence intervals rule. *Neural Networks*, 24(8):791–800, 2011.

[10] Cesare Alippi, Giacomo Boracchi, and Manuel Roveri. Just-in-time classifiers for recurrent concepts. *IEEE transactions on neural networks and learning systems*, 24(4):620–634, 2013.

[11] Pramod Anantharam, Payam Barnaghi, Krishnaprasad Thirunarayan, and Amit Sheth. Extracting city traffic events from social streams. *ACM Transactions on Intelligent Systems and Technology*, 9(4), 2014.

[12] Elena Andreou and Eric Ghysels. Structural breaks in financial time series. *Handbook of financial time series*, pages 839–870, 2009.

[13] Muhammad Tayyab Asif, Justin Dauwels, Chong Yang Goh, Ali Oran, Ehsan Fathi, Muye Xu, Menoth Mohan Dhanya, Nikola Mitrovic, and Patrick Jaillet. Spatiotemporal patterns in large-scale traffic speed prediction. *Intelligent Transportation Systems, IEEE Transactions on*, 15(2):794–804, 2014.

[14] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[15] A Aw and Michel Rascle. Resurrection of" second order" models of traffic flow. *SIAM journal on applied mathematics*, 60(3):916–938, 2000.

[16] P Bagnerini and M Rascle. A multiclass homogenized hyperbolic model of traffic flow. *SIAM journal on mathematical analysis*, 35(4):949–973, 2003.

[17] Masako Bando, Katsuya Hasebe, Ken Nakanishi, and Akihiro Nakayama. Analysis of optimal velocity model with explicit delay. *Physical Review E*, 58(5):5429, 1998.

[18] Masako Bando, Katsuya Hasebe, Akihiro Nakayama, Akihiro Shibata, and Yuki Sugiyama. Dynamical model of traffic congestion and numerical simulation. *Physical Review E*, 51(2):1035, 1995.

[19] N. Barimani, A.R. Kian, and B. Moshiri. Real time adaptive non-linear estimator/predictor design for traffic systems with inadequate detectors. *Intelligent Transport Systems, IET*, 8(3):308–321, May 2014.

[20] István Berkes, Edit Gombay, and Lajos Horváth. Testing for changes in the covariance structure of linear processes. *Journal of Statistical Planning and Inference*, 139(6):2044–2063, 2009.

[21] Gianluca Bontempi, Souhaib Ben Taieb, and Yann-Aël Le Borgne. Machine learning strategies for time series forecasting. In *European Business Intelligence Summer School*, pages 62–77. Springer, 2012.

[22] Olivier Bousquet and Léon Bottou. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168, 2008.

[23] B Bowerman, J Braverman, J Taylor, H Todosow, and U Von Wimmersperg. The vision of a smart city. In *2nd International Life Extension Technology Workshop, Paris*, volume 28, 2000.

[24] Mark Brackstone and Mike McDonald. Car-following: a historical review. *Transportation Research Part F: Traffic Psychology and Behaviour*, 2(4):181–196, 1999.

[25] Andrea Caragliu, Chiara Del Bo, and Peter Nijkamp. Smart cities in europe. *Journal of urban technology*, 18(2):65–82, 2011.

[26] Miguel A Carreira-Perpinan and Geoffrey E Hinton. On contrastive divergence learning. In *Proceedings of the tenth international workshop on artificial intelligence and statistics*, pages 33–40. Citeseer, 2005.

[27] Federico Castanedo. A review of data fusion techniques. *The Scientific World Journal*, 2013:19 pages, 2013.

[28] Rodolfo C Cavalcante, Leandro L Minku, and Adriano LI Oliveira. Fedd: Feature extraction for explicit concept drift detection in time series. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 740–747. IEEE, 2016.

[29] Tianfeng Chai and Roland R Draxler. Root mean square error (rmse) or mean absolute error (mae)? *Geoscientific Model Development Discussions*, 7:1525–1534, 2014.

[30] Kit Yan Chan, Tharam S Dillon, Jaipal Singh, and Elizabeth Chang. Neural-network-based models for short-term traffic flow forecasting using a hybrid exponential smoothing and levenberg–marquardt algorithm. *Intelligent Transportation Systems, IEEE Transactions on*, 13(2):644–654, 2012.

[31] Srinivasa Ravi Chandra and Haitham Al-Deek. Predictions of freeway traffic speeds and volumes using vector autoregressive models. *Journal of Intelligent Transportation Systems*, 13(2):53–72, 2009.

[32] H. Chang, Y. Lee, B. Yoon, and S. Baek. Dynamic near-term traffic flow prediction: systemoriented approach based on past experiences. *Intelligent Transport Systems, IET*, 6(3):292–305, September 2012.

[33] Tang-Hsien Chang, Albert Y Chen, Chia-Wen Chang, and Chia-Hung Chueh. Traffic speed estimation through data fusion from heterogeneous sources for first response deployment. *Journal of Computing in Civil Engineering*, 28(6):04014018, 2014.

[34] Haeran Cho and Piotr Fryzlewicz. Multiscale and multilevel technique for consistent segmentation of nonstationary time series. *Statistica Sinica*, pages 207–229, 2012.

[35] Luis M Correia and Klaus Wünstel. Smart cities applications and requirements. *White Paper. Net*, 2011.

[36] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585, 2006.

[37] Hua Cui, Lingling Peng, Huansheng Song, Guofeng Wang, Jiancheng Li, Lu Guo, and Chao Yuan. Dempster-shafer multifeature fusion for pedestrian detection. *Advances in Mechanical Engineering*, 7(1):454058, 2015.

[38] Yong Deng, Felix TS Chan, Ying Wu, and Dong Wang. A new linguistic mcdm method based on multiple-criterion data fusion. *Expert Systems with Applications*, 38(6):6985–6993, 2011.

[39] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.

[40] Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in non-stationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015.

[41] S. Dunne and B. Ghosh. Weather adaptive traffic prediction using neurowavelet models. *Intelligent Transportation Systems, IEEE Transactions on*, 14(1):370–379, March 2013.

[42] Yogesh Dwivedi and Suhasini Subba Rao. A test for second-order stationarity of a time series based on the discrete fourier transform. *Journal of Time Series Analysis*, 32(1):68–91, 2011.

[43] Nour-Eddin El Faouzi, Lawrence Klein, and Olivier De Mouzon. Improving travel time estimates from inductive loop and toll collection data with dempster-shafer data fusion. *Transportation Research Record: Journal of the Transportation Research Board*, (2129):73–80, 2009.

[44] Ryan Elwell and Robi Polikar. Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, 22(10):1517–1531, 2011.

[45] Yang Fang, Wang Shangguang, Li Jinglin, Liu Zhihan, and Sun Qibo. An overview of internet of vehicles. *Communications, China*, 11(10):1–15, Oct 2014.

[46] Florentino Fdez-Riverola, Eva Lorenzo Iglesias, Fernando Díaz, José Ramon Méndez, and Juan M Corchado. Applying lazy learning algorithms to tackle concept drift in spam filtering. *Expert Systems with Applications*, 33(1):36–48, 2007.

[47] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

[48] Marco Fontani, Tiziano Bianchi, Alessia De Rosa, Alessandro Piva, and Mauro Barni. A framework for decision fusion in image forensics based on dempster–shafer theory of evidence. *Information Forensics and Security, IEEE Transactions on*, 8(4):593–607, 2013.

[49] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[50] Denos C Gazis, Robert Herman, and Richard W Rothery. Nonlinear follow-the-leader models of traffic flow. *Operations research*, 9(4):545–567, 1961.

[51] Rudolf Giffinger, Christian Fertner, Hans Kramar, Robert Kalasek, Nataša Pichler-Milanovic, and Evert Meijers. Smart cities-ranking of european medium-sized cities. Technical report, Vienna University of Technology, 2007.

[52] Peter G Gipps. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 15(2):105–111, 1981.

[53] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[54] Paulo Mauricio Gonçalves Jr and Roberto Souto Maior De Barros. Rcd: A recurring concept drift framework. *Pattern Recognition Letters*, 34(9):1018–1025, 2013.

[55] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[56] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[57] Bjørn Grung and Rolf Manne. Missing values in principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 42(1):125–139, 1998.

[58] Jose A Guajardo, Richard Weber, and Jaime Miranda. A model updating strategy for predicting time series with seasonal patterns. *Applied Soft Computing*, 10(1):276–283, 2010.

[59] Jianhua Guo and Billy Williams. Real-time short-term traffic speed level forecasting and uncertainty quantification using layered kalman filters. *Transportation Research Record: Journal of the Transportation Research Board*, 2175:28–37, 2010.

[60] D.L. Hall, M. McNeese, J. Llinas, and T. Mullen. A framework for dynamic hard/soft fusion. In *Information Fusion, 2008 11th International Conference on*, pages 1 – 8, 30 2008-july 3 2008.

[61] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*, 2016.

[62] Hany M Hassan and Mohamed A Abdel-Aty. Predicting reduced visibility related crashes on freeways using real-time traffic flow data. *Journal of safety research*, 45:29–36, 2013.

[63] Jingrui He, Wei Shen, Phani Divakaruni, Laura Wynter, and Rick Lawrence. Improving traffic prediction with tweet semantics. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 1387–1393. AAAI Press, 2013.

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

[65] Dirk Helbing. Modeling multi-lane traffic flow with queuing effects. *Physica A: Statistical Mechanics and its Applications*, 242(1):175–194, 1997.

[66] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[67] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[68] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[69] Serge P Hoogendoorn and Piet HL Bovy. Generic gas-kinetic traffic systems modeling with applications to vehicular traffic flow. *Transportation Research Part B: Methodological*, 35(4):317–336, 2001.

[70] Serge P Hoogendoorn, Hans van Lint, and Victor Knoop. Dynamic first-order modeling of phase-transition probabilities. In *Traffic and Granular Flow07*, pages 85–92. Springer, 2009.

[71] Jianming Hu, Pan Gao, Yunfei Yao, and Xudong Xie. Traffic flow forecasting with particle swarm optimization and support vector regression. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 2267–2268, Oct 2014.

[72] Wenhao Huang, Guojie Song, Haikun Hong, and Kunqing Xie. Deep architecture for traffic flow prediction: Deep belief networks with multitask learning. *Intelligent Transportation Systems, IEEE Transactions on*, 15(5):2191–2201, Oct 2014.

[73] Zhou Jiang, Cunbao Zhang, and Yinxia Xia. Travel time prediction model for urban road network based on multi-source data. *Procedia - Social and Behavioral Sciences*, 138(0):811 – 818, 2014. The 9th International Conference on Traffic and Transportation Studies (ICTTS 2014).

[74] Sheng Jin, Dian-hai Wang, Cheng Xu, and Dong-fang Ma. Short-term traffic safety forecasting using gaussian mixture model and kalman filter. *Journal of Zhejiang University SCIENCE A*, 14(4):231–243, 2013.

[75] Wen-Long Jin. A kinematic wave theory of lane-changing traffic flow. *Transportation research part B: methodological*, 44(8):1001–1021, 2010.

[76] Bahador Khaleghi, Alaa Khamis, Fakhreddine O. Karray, and Saiedeh N. Razavi. Multisensor data fusion: A review of the state-of-the-art. *Information Fusion*, 14(1):28 – 44, 2013.

[77] Bahador Khaleghi, Alaa Khamis, Fakhreddine O Karray, and Saiedeh N Razavi. Multisensor data fusion: A review of the state-of-the-art. *Information Fusion*, 14(1):28–44, 2013.

[78] Meghdad Khazaee, Hojat Ahmadi, Mahmoud Omid, Ashkan Moosavian, and Majid Khazaee. Classifier fusion of vibration and acoustic signals for fault diagnosis and classification of planetary gears based on dempster–shafer evidence theory. *Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering*, 228(1):21–32, 2014.

[79] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[80] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, page 201611835, 2017.

[81] Wolfgang Knospe, Ludger Santen, Andreas Schadschneider, and Michael Schreckenberg. Empirical test for cellular automaton models of traffic flow. *Physical Review E*, 70(1):016115, 2004.

[82] Arief Koesdwiady, Ridha Soua, and Fakhreddine Karray. Improving traffic flow prediction with weather information in connected cars: A deep learning approach. *IEEE Transactions on Vehicular Technology*, 65(12):9508–9517, 2016.

[83] Ben Kolosz, Susan Grant-Muller, and Karim Djemame. Modelling uncertainty in the sustainability of intelligent transport systems for highways using probabilistic data fusion. *Environmental Modelling & Software*, 49:78–97, 2013.

[84] J Zico Kolter and Marcus A Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, 8(Dec):2755–2790, 2007.

[85] Karolos K Korkas and Piotr Fryzlewicz. Multiple change-point detection for non-stationary time series using wild binary segmentation. *Statistica Sinica*, 27(1):287–311, 2017.

[86] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[87] Paolo La Greca, Luca Barbarossa, Matteo Ignaccolo, Giuseppe Inturri, and Francesco Martinico. The density dilemma. a proposal for introducing smart growth principles in a sprawling settlement within catania metropolitan area. *Cities*, 28(6):527–535, 2011.

[88] Jorge A Laval and Ludovic Leclercq. A mechanism to describe the formation and propagation of stop-and-go waves in congested freeway traffic. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 368(1928):4519–4541, 2010.

[89] JP Lebacque. A two phase extension of the lrw model based on the boundedness of traffic acceleration. In *Transportation and traffic theory in the 21st century. Proceedings of the 15th international symposium on transportation and traffic theory*, 2002.

[90] Ludovic Leclercq. Hybrid approaches to the solutions of the lighthill–whitham–richards model. *Transportation Research Part B: Methodological*, 41(7):701–709, 2007.

[91] Ludovic Leclercq. A new numerical scheme for bounding acceleration in the lwr model. In *4th IMA International Conference on Mathematics in Transport*, 2007.

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

[93] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616. ACM, 2009.

[94] Michael J Lighthill and Gerald Beresford Whitham. On kinematic waves. ii. a theory of traffic flow on long crowded roads. In *Proceedings of the Royal Society of London*

A: Mathematical, Physical and Engineering Sciences, volume 229, pages 317–345. The Royal Society, 1955.

[95] Lei Lin, Ming Ni, Qing He, Jing Gao, Adel W Sadek, and Transportation Informatics Tier I Director. Modeling the impacts of inclement weather on freeway traffic speed: An exploratory study utilizing social media data. *to appear in Transportation Research Record: Journal of Transportation Research Board*, 2015.

[96] Van Lint and Van Hinsbergen. Short-term traffic and travel time prediction models. *Artif. Intell. Appl. Crit. Transp*, pages 22–41, 2012.

[97] Song Liu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013.

[98] Hua-pu Lu, Zhi-yuan Sun, and Wen-cong Qu. Big data-driven based real-time traffic flow state identification and prediction. *Discrete Dynamics in Nature and Society*, 2014.

[99] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic flow prediction with big data: A deep learning approach. *Intelligent Transportation Systems, IEEE Transactions on*, 16(2):865–873, April 2015.

[100] Brian Maguire and Shaishav Desai. Dempster-shafer fusion for personnel detection: Application of dempster-shafer theory with ultrasonic micro-doppler and pir sensors. In *Information Fusion (FUSION), 2012 15th International Conference on*, pages 2209–2214. IEEE, 2012.

[101] Highway Capacity Manual. Highway capacity manual. *Washington, DC*, 2000.

[102] Highway Capacity Manual. Volumes 1-4.(2010). *Transporation Research Board*, 2010.

[103] Massimiliano Marcellino, James H Stock, and Mark W Watson. A comparison of direct and iterated multistep ar methods for forecasting macroeconomic time series. *Journal of econometrics*, 135(1):499–526, 2006.

[104] Iván Marsá-Maestre, Miguel A López-Carmona, Juan R Velasco, and Andrés Navarro. Mobile agents for service personalization in smart environments. *Journal of Networks*, 3(5):30–41, 2008.

[105] Jennifer McCrea and Salissou Moutari. A hybrid macroscopic-based model for traffic flow in road networks. *European Journal of Operational Research*, 207(2):676 – 684, 2010.

[106] Wanli Min and Laura Wynter. Real-time road traffic prediction with spatio-temporal correlations. *Transportation Research Part C: Emerging Technologies*, 19(4):606 – 616, 2011.

[107] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[108] Abdel-rahman Mohamed, Tara N Sainath, George Dahl, Bhuvana Ramabhadran, Geoffrey E Hinton, Michael Picheny, et al. Deep belief networks using discriminative features for phone recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5060–5063. IEEE, 2011.

[109] Luis Moreira-Matias, João Gama, and João Mendes-Moreira. Concept neurons–handling drift issues for real-time industrial data mining. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 96–111. Springer, 2016.

[110] Jesús Muñuzuri, Juan Larrañeta, Luis Onieva, and Pablo Cortés. Solutions applicable by local administrations for urban logistics improvement. *Cities*, 22(1):15–28, 2005.

[111] Kai Nagel and Michael Schreckenberg. A cellular automaton model for freeway traffic. *Journal de physique I*, 2(12):2221–2229, 1992.

[112] Vinod Nair and Geoffrey E Hinton. 3d object recognition with deep belief nets. In *Advances in Neural Information Processing Systems*, pages 1339–1347, 2009.

[113] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[114] Taewoo Nam and Theresa A Pardo. Conceptualizing smart city with dimensions of technology, people, and institutions. In *Proceedings of the 12th Annual International Digital Government Research Conference: Digital Government Innovation in Challenging Times*, pages 282–291. ACM, 2011.

[115] Liu Nanjie. Internet of Vehicles Your next connection. *WinWin*, 2011.

[116] Guy P Nason. Stationary and non-stationary time series. *Statistics in Volcanology. Special Publications of IAVCEI*, 1:000–000, 2006.

[117] Paolo Neirotti, Alberto De Marco, Anna Corinna Cagliano, Giulio Mangano, and Francesco Scorrano. Current trends in smart city initiatives: Some stylised facts. *Cities*, 38:25–36, 2014.

[118] Gordon Frank Newell. Nonlinear effects in the dynamics of car following. *Operations research*, 9(2):209–229, 1961.

[119] Gordon Frank Newell. A simplified car-following theory: a lower order model. *Transportation Research Part B: Methodological*, 36(3):195–205, 2002.

[120] Luis Leon Ojeda, Alain Y Kibangou, and Carlos Canudas De Wit. Adaptive kalman filtering for multi-step ahead traffic flow prediction. In *American Control Conference (ACC), 2013*, pages 4724–4729. IEEE, 2013.

[121] Louis A Pipes. An operational analysis of traffic dynamics. *Journal of applied physics*, 24(3):274–281, 1953.

[122] Robi Polikar, Lalita Upda, Satish S Upda, and Vasant Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, 31(4):497–508, 2001.

[123] MB Priestley and T Subba Rao. A test for non-stationarity of time-series. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 140–149, 1969.

[124] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

[125] Yalda Rajabzadeh, Amir Hossein Rezaie, and Hamidreza Amindavar. Short-term traffic flow prediction using time-varying vasicek model. *Transportation Research Part C: Emerging Technologies*, 74:168–181, 2017.

[126] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.

[127] Hubert Rehborn and Micha Koller. A study of the influence of severe environmental conditions on common traffic congestion features. *Journal of Advanced Transportation*, 48(8):1107–1120, 2014.

[128] Chuan-xiang Ren, Cheng-bao Wang, Chang-chang Yin, Meng Chen, and Xu Shan. The prediction of short-term traffic flow based on the niche genetic algorithm and bp neural network. In *Proceedings of the 2012 International Conference on Information Technology and Software Engineering*, pages 775–781. Springer, 2013.

[129] Mahdi Rezaei and Mutsuhiro Terauchi. Vehicle detection based on multi-feature clues and dempster-shafer fusion theory. In *Image and Video Technology*, pages 60–72. Springer, 2014.

[130] Paul I Richards. Shock waves on the highway. *Operations research*, 4(1):42–51, 1956.

[131] Gordon J Ross, Niall M Adams, Dimitris K Tasoulis, and David J Hand. Exponentially weighted moving average charts for detecting concept drift. *Pattern Recognition Letters*, 33(2):191–198, 2012.

[132] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

[133] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[134] Nicholas Sapankevych, Ravi Sankar, et al. Time series prediction using support vector machines: a survey. *Computational Intelligence Magazine, IEEE*, 4(2):24–38, 2009.

[135] Glenn Shafer et al. *A mathematical theory of evidence*, volume 1. Princeton university press Princeton, 1976.

[136] Lingling Song. Improved intelligent method for traffic flow prediction based on artificial neural networks and ant colony optimization. *Journal of Convergence Information Technology*, 7(8), 2012.

[137] Ridha Soua, Arief Koesdwiady, and Fakhri Karray. Big-data-generated traffic flow prediction using deep learning and dempster-shafer theory. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 3195–3202. IEEE, 2016.

[138] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[139] Souhaib Ben Taieb. *Machine learning strategies for multi-step-ahead time series forecasting*. PhD thesis, Universit Libre de Bruxelles, Belgium, 2014.

[140] Souhaib Ben Taieb and Amir F Atiya. A bias and variance analysis for multistep-ahead time series forecasting. *IEEE transactions on neural networks and learning systems*, 27(1):62–76, 2016.

[141] Souhaib Ben Taieb, Gianluca Bontempi, Amir F Atiya, and Antti Sorjamaa. A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. *Expert systems with applications*, 39(8):7067–7083, 2012.

[142] Souhaib Ben Taieb, Rob J Hyndman, et al. Recursive and direct multi-step forecasting: the best of both worlds. *Monash University, Department of Econometrics and Business Statistics, Tech. Rep*, 2012.

[143] Chris Tampère, Bart Van Arem, and S Hoogendoorn. Gas-kinetic traffic flow modeling including continuous driver behavior models. *Transportation Research Record: Journal of the Transportation Research Board*, (1852):231–238, 2003.

[144] Tom Thomas, Wendy Weijermars, and Eric Van Berkum. Predictions of urban volumes in single time series. *Intelligent Transportation Systems, IEEE Transactions on*, 11(1):71–80, 2010.

[145] Donato Toppeta. The smart city vision: How innovation and ict can build smart," liveable", sustainable cities. *The Innovation Knowledge Foundation. Think*, 2010.

[146] Martin Treiber and Arne Kesting. Traffic flow dynamics. *Traffic Flow Dynamics: Data, Models and Simulation, Springer-Verlag Berlin Heidelberg*, 2013.

[147] Martin Treiber, Arne Kesting, and Dirk Helbing. Three-phase traffic theory and two-phase models with a fundamental diagram in the light of empirical stylized facts. *Transportation Research Part B: Methodological*, 44(8):983–1000, 2010.

[148] Ioannis Tsapakis, Tao Cheng, and Adel Bolbol. Impact of weather conditions on macroscopic urban travel times. *Journal of Transport Geography*, 28:204 – 211, 2013.

[149] Femke van Wageningen-Kessels, Hans van Lint, Serge Hoogendoorn, and Kees Vuik. Lagrangian formulation of multiclass kinematic wave model. *Transportation Research Record: Journal of the Transportation Research Board*, (2188):29–36, 2010.

[150] Femke van Wageningen-Kessels, Bas van't Hof, Serge P Hoogendoorn, Hans Van Lint, and Kees Vuik. Anisotropy in generic multi-class traffic flow models. *Transportmetrica A: Transport Science*, 9(5):451–472, 2013.

[151] Arun Venkatraman, Martial Hebert, and J Andrew Bagnell. Improving multi-step prediction of learned time series models. In *AAAI*, pages 3024–3030, 2015.

[152] EleniI. Vlahogianni and MatthewG. Karlaftis. Comparing traffic flow time-series under fine and adverse weather conditions using recurrence-based complexity measures. *Nonlinear Dynamics*, 69(4):1949–1963, 2012.

[153] Chenqi Wang and Hsin-Mu Tsai. Detecting urban traffic congestion with single vehicle. In *Connected Vehicles and Expo (ICCVE), 2013 International Conference on*, pages 233–240. IEEE, 2013.

[154] Dong Wang, Jie Xiong, Zhu Xiao, and Xiaohong Li. Short-term traffic flow prediction based on ensemble real-time sequential extreme learning machine under non-stationary condition. In *Vehicular Technology Conference (VTC Spring), 2016 IEEE 83rd*, pages 1–5. IEEE, 2016.

[155] Jin Wang and Qixin Shi. Short-term traffic speed forecasting hybrid model based on chaos–wavelet analysis-support vector machine theory. *Transportation Research Part C: Emerging Technologies*, 27:219–232, 2013.

[156] Doug Washburn, Usman Sindhu, S Balaouras, RA Dines, N Hayes, and LE Nelson. Helping cios understand smart city initiatives. *Growth*, 17, 2009.

[157] B Williams and L Hoel. Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. *Journal of Transportation Engineering*.

[158] Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005.

[159] Tianshu Wu, Kunqing Xie, Dong Xinpin, and Guojie Song. A online boosting approach for traffic flow forecasting under abnormal conditions. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, pages 2555–2559, May 2012.

[160] Koichi Yamada, Vilany Kimala, and Muneyuki Unehara. A new conditioning rule, its generalization and evidential reasoning. In *IFSA/EUSFLAT Conf.*, pages 92–97, 2009.

[161] Hongbin Yin, S_C Wong, Jianmin Xu, and CK Wong. Urban traffic flow prediction using a fuzzy-neural approach. *Transportation Research Part C: Emerging Technologies*, 10(2):85–98, 2002.

[162] Donggeun Yoo, Namil Kim, Sunggyun Park, Anthony S Paek, and In So Kweon. Pixel-level domain transfer. In *European Conference on Computer Vision*, pages 517–532. Springer, 2016.

[163] Bin Yu, Xiaolin Song, Feng Guan, Zhiming Yang, and Baozhen Yao. k-nearest neighbor model for multiple-time-step prediction of short-term traffic condition. *Journal of Transportation Engineering*, 142(6):04016018, 2016.

[164] Haiyang Yu, Zhihai Wu, Shuqin Wang, Yunpeng Wang, and Xiaolei Ma. Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks. *Sensors*, 17(7):1501, 2017.

[165] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. Internet of things for smart cities. *Internet of Things Journal, IEEE*, 1(1):22–32, Feb 2014.

[166] Matthew D Zeiler, M Ranzato, Rajat Monga, Min Mao, Kun Yang, Quoc Viet Le, Patrick Nguyen, Alan Senior, Vincent Vanhoucke, Jeffrey Dean, et al. On rectified linear units for speech processing. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3517–3521. IEEE, 2013.

[167] Ping Zhang, Kunqing Xie, and Guojie Song. A short-term freeway traffic flow prediction method based on road section traffic flow structure pattern. In *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pages 534–539. IEEE, 2012.

[168] Yanru Zhang, Yunlong Zhang, and Ali Haghani. A hybrid short-term traffic flow forecasting method based on spectral analysis and statistical volatility model. *Transportation Research Part C: Emerging Technologies*, 43, Part 1:65 – 78, 2014. Special Issue on Short-term Traffic Flow Forecasting.

[169] Yunjie Zhao, Adel Sadek, and Daniel Fuglewicz. Modeling the impact of inclement weather on freeway traffic speed at macroscopic and microscopic levels. *Transportation Research Record: Journal of the Transportation Research Board*, 2272:173–180, 2012.

[170] Weizhong Zheng, Der-Horng Lee, and Qixin Shi. Short-term freeway traffic flow prediction: Bayesian combined neural network approach. *Journal of transportation engineering*, 132(2):114–121, 2006.