Mobile Traffic Prediction Through Machine Learning Techniques

Fuyou Li

Department of Electronic and Electrical Engineering

University of Sheffield

Supervisors: Prof. Jie Zhang, Dr. Wei Liu

This thesis is submitted for the approval of the

Doctor of Philosophy

July 2021

This thesis is dedicated to my beloved parents. Without their unconditional love and support, I would not be who I am today.

Acknowledgements

First and foremost, I would like to express my sincere thanks to my supervisor, Professor Jie Zhang, for his continuous guide and support during my PhD study. His patience, encouragement and guidance help me to complete the study. His optimism and confidence towards life and work have a positive impact on me. Besides my supervisor, I would like to thank my second supervisor, Dr Wei Liu, for giving me great support.

I am very grateful to Dr. Zitian Zhang who most importantly helped me to become an independent researcher. I could never accomplish my PhD study without his guidance and persistent help.

A very special gratitude to Dr. Yunpeng Zhu and Mr. Lu Zhang for their guidance and support.

Finally, I would like to express my heartfelt appreciation for my family who always encourages and support me with patience and love.

Abstract

In recent decades, the development of wireless technologies incurs explosive mobile traffic growth. To address the rapidly growing traffic demand, operators deploy more base stations to increase the total network traffic capacity. However, the deployment significantly increases the operational cost, and the traffic demand is still hard to be fulfilled. Besides, more mobile applications and services rely on nearly real-time or even proactive traffic analysis. These increasing traffic demands and increasingly stringent quality of service requirements have brought significant challenges. Mobile traffic analysis becomes a promising solution to these challenges and attracts continuous research interest from both academia and industry. Therefore, mobile traffic analysis is the main research direction of this thesis.

Firstly, user mobility analysis, a critical perspective in mobile traffic analysis, is conducted. Long Short-Term Memory (LSTM) network, an elegant candidate to address this issue, is introduced and investigated. An LSTM-based user mobility prediction scheme is proposed and evaluated through a real-world user trajectory dataset.

Next, the thesis focuses on mobile network traffic pattern analysis. Twitter traffic is analysed to extract the temporal characteristics. Based on the extracted features, a Twitter traffic prediction framework is proposed which combines statistical analysis and machine learning techniques.

After that, this thesis seeks to improve mobile network traffic prediction accuracy. Although recent research shows the potentials of deep learning-based algorithms, current techniques have high complexity, and require a long time and a high volume of samples to train the model. Therefore, a meta-learning-based mobile traffic prediction framework, ML-

TP, is proposed to address these issues, achieving higher prediction accuracy. Furthermore, the learning efficiency is also significantly improved.

Finally, to further improve ML-TP, dmTP is proposed, determining the optimal model hyperparameter. The dmTP shows competitive performance in both regular and high varied traffic prediction accuracy and learning efficiency by testing several real-world datasets.

List of Publications

Published

F. Li, Z. Zhang, Y. Zhu and J. Zhang, "Prediction of Twitter Traffic Based on Machine Learning and Data Analytics," IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, 2020, pp. 443-448

Z. Zhang, F. Li, X. Chu and J. Zhang, "The dmTP: A Deep Meta-learning based Framework for Mobile Traffic Prediction," accepted by IEEE Wireless Communications Magazine, 2021

Submitted

F. Li, Z. Zhang, X. Chu and J. Zhang, " A Meta-Learning based Framework for Cellular Level Mobile Network Traffic Prediction," submitted to IEEE Transactions on Wireless Communications

List of Abbreviations

3GPP	3rd Generation Partnership Project
ANN	Artificial Neural Network
ARIMA	AutoRegressive Integrated Moving Average
ATT	Average Training Time
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DFS	Discrete Fourier Series
dmTP	deep meta-learning based mobile Traffic Prediction framework
FFT	Fast Fourier Transform
IFFT	Inverse Fast Fourier Transform
KNN	K-nearest neighbours
LTE	Long Term Evolution
LR	Linear Regression
LSTM	Long Short-Term Memory
MLP	Multilayer Perceptron
ML-TP	Meta-Learning Traffic Prediction
NMAE	Normalised Mean Absolute Error
NMSE	Normalised Mean Squared Error
NRMSE	Normalised Root Mean Square Error
OpEx	Operational Expenditure
OSN	Online Social Network
POI	Point of Interest
R2	R-squared
RNN	Recurrent Neural Networks
SVR	Support Vector Regression
TCMTM	Time needed to Construct the Meta-task training set and Train the Meta-learner

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Background and Motivation

1.1.1 The Development of Mobile Network Traffic Analysis

Mobile traffic analysis becomes a popular research topic in recent years [1]. The early, seminal studies exploiting mobile network traffic datasets of significant size appeared in 2006 [2]. Although this research field is relatively new, it develops rapidly. The reasons behind this can be summarised into four factors. Unprecedented coverage of mobile services is the first factor. Mobile services are becoming more popular and ubiquitous. According to statistics from the Global System for Mobile Communications Association (GSMA), the number of global mobile scribers has exceeded 5.2 billion in 2020 [3]. This means that about two-thirds of the global population has at least one mobile device.

As a consequence, the mobile scribers can represent the population to a large extent [4]. Secondly, the continuous interactions between mobile devices and the network infrastructure produce vast subscribers' information at unprecedented scales, such as movement, interactions, service usage, and other users' behaviour-related information [1]. This rich source of knowledge attracts much attention from experts from different disciplines, such as sociology, epidemiology, transportation, and wireless communications [1]. The third supporting factor is the availability of datasets. Although mobile operators keep monitoring

the mobile traffic status in their networks to troubleshoot and improve the quality of service and billing [1], they barely share the data. This situation is changing since they see the potential benefits of sharing data and collaborations with research groups. The last factor is the quality improvement of datasets. Due to the base station densification and diversity of services, from macro base stations to small base stations, from voice and texting to data and diverse applications, the granularity, diversity, and accuracy of data are significantly improved [1].

Since the mobile traffic data have diverse types of information, and the research is conducted from various disciplines, it is difficult to find a clear edge between each research domains [1]. One way is to classify the research according to the research subjects. Based on the research domains, the research can be classified into three subjects: social analyses, mobility analyses, and network analyses [1]. The social analysis focus on exploring user interactions [2, 5], demographics [6, 7], and environmental perspectives such as epidemics prevention and containment [8–11]. The mobility analysis extract either global or individual movement patterns [12–32]. The network analysis aims to understand mobile traffic dynamics to improve mobile network performance [1]. Basically, in wireless communications, the mobile network traffic analysis covers all these three subjects, therefore plays a vital role in the field. For example, from the social perspective, Li et al. [33] propose a social-aware device-to-device communication architecture; in the field of mobility analysis, many works deal with the mobility aware computation or content offloading strategies in mobile networks [34–36]; from the network performance point of view, Hu et al. [37] propose a base station sleeping strategy to reduce the energy consumption based on mobile traffic prediction.

1.1.2 Machine Learning in Mobile Networking

Machine learning was first proposed in 1952, which can be defined as a study of computer algorithms that can make predictions or improvements through experience [38]. Machine learning aims to utilise the data to build a model that can make predictions or decisions automatically [38]. Compared with traditional mathematical models, machine learning-based models require considerable data to train the model rather than explicitly programmed [38].

By developing its algorithm, machine learning can solve complicated problems that are difficult for humans to build appropriate models manually [37]. Based on the model of the brain and machine learning concept in [38], Rosenblatt [39] created the perceptron. Although the perceptron seemed promising, Minsky and Papert [40] proved that it could only deal with linear separable problems. Due to this defect of the neural network and limited computing power and pessimism about machine learning effectiveness, the development of machine learning has stalled. The resurgence of machine learning research occurred in the 1980s, when the backpropagation theory was proposed [41]. In the 1990s, machine learning research shifts from a knowledge-driven approach to a data-driven approach [42]. Support vector machines [43] and Recurrent Neural Networks (RNNs) [44] were proposed, and the computational complexity of neural networks increases. From the 2010s, deep learning becomes feasible, making machine learning a part of many software services and applications [45]. Examples of deep learning architectures include deep neural networks, deep belief networks, and convolutional neural networks.

Machine learning can be mainly classified into two categories: supervised learning and unsupervised learning. A supervised learning task has an input-output pair, meaning that it has both an input object and the desired output value. It aims to learn the function which maps the input-output pairs [46]. Since each sample in supervised learning is tagged with the desired output value (label), the training process is easy to understand. The evaluation of the accuracy of supervised learning algorithms is relatively straightforward. Most regression and classification problems are supervised learning tasks. Typical examples of supervised learning include handwriting recognition, spam detection, pattern recognition, and speech recognition.

In contrast to supervised learning, unsupervised learning does not have pre-existing labels [46]. It aims to identify the patterns or relationships which are previously unknown. Compared with supervised learning who has targets, unsupervised learning must discover information without any guidance or supervision. Hence it is more unpredictable. The advantage of unsupervised learning is that it can perform more complex processing tasks than supervised learning [46]. The most common unsupervised learning applications in-

clude clustering, anomaly detection [47], and feature extraction. In mobile networks, both supervised and unsupervised learning algorithms are involved [48], and supervised learning is more popular since the targets are usually available under more scenarios, such as user locations in mobility management and traffic loads in network optimisation problems.

According to the state of the arts, machine learning algorithms and mobile networking are two independent fields. Some crossovers emerged in recent years indicate that machine learning techniques start to be employed to solve complicated problems in mobile networks, especially in the 5th Generation (5G) of mobile systems where things become much more complex, such as radio resource management [49], mobility management [50] and mobile traffic prediction [51, 52]. With mobile communication technology development, the diversity and complexity of mobile networks increase significantly, causing the traditional network management techniques to become challenging to meet the needs [47]. For example, in the 3G, wireless coverage optimisation needs more than one hour to apply [53]. Although things have been improved in the 4G networks, the advanced self-optimisation network engines still spend more than ten minutes to react [54, 55]. However, in the 5G, the context in a mobile network, such as the number of users in a region, changes fast and frequently. Hence, fast-response and proactive optimisation become a necessity [56]. The 3rd Generation Partnership Project (3GPP) Mobile Data Applications Impacts (Release 11) [57] has mentioned that the network optimisation can be boosted if the user behaviour and mobile network dynamics can be understood and predicted. Machine learning techniques show great potentials in extracting user behaviour and spatial-temporal traffic patterns [58]. Researchers recognise the importance and potential of using machine-learning techniques to solve problems in mobile networks [59, 60]. Li et al. [61] discuss the potentials and effectiveness of integrating machine learning into mobile networks to achieve intelligent 5G networks and highlight the significance of machine learning in future mobile network architectures. Therefore, integrating machine learning technology into mobile networks becomes a promising research field.

1.1.3 Motivation

With the rapid development of cellular network technologies and mobile applications' innovations, mobile networks' traffic has increased exponentially in recent decades [62]. The number of new devices added in 2016 reached nearly half a billion [63]. The global mobile data traffic has increased by eighteen-fold between 2012 and 2016. The global mobile data traffic volume has reached 19.01 exabytes (10_{18} bytes) per month and will increase at a 46 % annual growth rate [64]. Mobile traffic is expected to has an astounding 1000-fold increase in this decade [65].

To meet these user demands and reduce the Operational Expenditure (OpEx), mobile traffic prediction plays a vital role in mobile networks since many mobile applications rely on real-time or approximately real-time traffic analysis throughout a considered Radio Access Network (RAN) [37, 66, 67]. For example, operators have to deploy more base stations to meet the rapidly growing user demands [68]. Although the power consumption of a small cell is lower than that of a macro base station, the increase in the number of base stations makes the total power consumption increase. By forecasting the mobile traffic, a sleeping strategy for the BS might be carried out to reduce energy consumption, which accounts for a significant proportion of OpEx [37]. The 3GPP Release 8 proposed automatic optimisation in mobile networks and introduced and self-optimisation network [67]. For congestion control, a key element of keeping the quality of service in a self-optimisation network, accurate traffic models can capture the actual statistical traffic characteristics in the network. An inaccurate traffic prediction model either over-estimates or under-estimates the network traffic will degrades network performance [69]. Besides, with the introduction of network slicing, such as network virtualisation and artificial intelligence, the 5G and future mobile networks will operate in an ultra-flexible way [70, 71]. The number of slices that can be run on the same infrastructure determines the network performance, relying on traffic prediction ability [72]. Therefore, mobile network traffic prediction becomes more critical in network deployment, management, optimisation, improving the users' quality of experience, and even acquiring intelligence [61, 73, 74]. This is why mobile traffic prediction has become a hot

topic attracting attention from both academia and industries in recent years. Meanwhile, the nature of mobile network traffic makes prediction a complicated and challenging task.

As introduced above, many efforts have been made to predict mobile network traffic. Numerous methods have been proposed from statistical, mathematical to machine learning perspectives, and significant progress has been achieved. The early works aim to discover mobile network traffic features, such as periodicity [75] and correlations [12]. Their findings lay the foundations for predicting mobile network traffic. Later research starts to try using mathematical models to model the mobile network traffic dynamics, such as the Markov model [76], α-stable model [77], Holt-Winter's exponential smoothing model [78] and autoregressive moving average model [79]. Although these approaches try to depict mobile network traffic mathematically, the accuracy is still unsatisfactory. The emergence of machine learning techniques makes further improvements possible. Machine learning techniques show the potentials of achieving higher prediction accuracy without complex mathematical models. Such techniques include Linear Regression (LR) [80], compressive sensing [81], Support Vector Regression (SVR) [82], ANN [83] and RNN [84]. However, the improvements come at a cost. Compared with mathematical-based models, machine learning-based approaches require more data to train the model, along with longer model training time and higher computation complexity.

Nevertheless, some challenges are brought by the nature of the mobile network traffic dynamics, which are inevitable and worthy of attention. Meanwhile, the limitations of current works should also be noticed. Some of the main challenges and limitations are summarised as follows.

- The accuracy of mobile traffic prediction is a necessity [72]. Mobile network traffic is very complicated because of many factors, such as time, day of the week, number of users, and user behaviours. In order to capture mobile traffic dynamics, an accurate prediction model is required.

- Existing mobile network traffic prediction models based on mathematical methods simply simulate mobile network traffic variation, which ignores the features and correlations hidden in the traffic, such as the periodicity and temporal correlations

[80]. These correlations help to understand user behaviour and design mobile network optimisation strategy.

- Although the machine learning-based methods are proposed to improve significantly the ability to capture traffic dynamics, these methods have significant computation complexity. They require more time to train the model than those based on mathematical methods [78, 85–89]. An efficient mobile network prediction method is needed.

In order to address the challenges, this thesis explores a traffic pre-processing method to improve the mobile traffic prediction accuracy and develop a deep-learning-based mobile traffic prediction framework that achieves competitive prediction performance with high computation efficiency.

1.2 Contributions of the Thesis

The main contributions of this thesis are summarised as follows:

- Propose a feature extraction method that combines both statistical analysis and frequency domain analysis. It can extract daily traffic pattern features and filter out the outliers of mobile network traffic.

- Introduce a mobile traffic prediction framework using the proposed feature extraction technique to achieve better prediction accuracy.

- Develop a novel meta-learning method in the mobile traffic prediction framework, such to achieve better prediction performance, which is justified by the Normalised Root Mean Square Error (NRMSE), Normalised Mean Absolute Error (NMAE), and R-squared (R2) criteria. Simultaneously, the proposed framework dramatically improves training efficiency in terms of training time and computation load consumption.

- Propose an advanced meta-learning-based mobile network traffic prediction framework that utilises a Multilayer Perceptron (MLP) as the meta-learner. This framework

can determine not only the model parameters of the base-learners but also the hyper-parameters. By introducing the meta-learner, the baseline methods' prediction accuracy can also be improved. The proposed framework also shows competitive performance in predicting the traffic load from another city with different spatial scales and different types of services.

- Practical applications of the newly developed approaches are demonstrated by using the real-world mobile network traffic datasets from Italy, London, and China, respectively. These approaches show the potentials of solving problems in practice.

1.3 Thesis Organisation

The thesis is organised as follows.

Chapter 2: Literature Review

This chapter presents the development and related works related to mobile network traffic prediction. Then, the machine learning techniques applied in mobile communication networks are introduced.

Chapter 3: Fundamentals of LSTM

This chapter focuses on introducing the LSTM network, including the principles of LSTM and how it is used in mobile networks. This lays the foundation for its application in mobile network traffic prediction. A use case of applying LSTM in mobile networks is illustrated.

Chapter 4: Traffic Feature Analysis

In this chapter, the features of mobile network traffic are analysed. The analysis is conducted in both the time domain and frequency domain to extract mobile network traffic features. This chapter proposes a Twitter traffic prediction framework that combines the proposed analysis technique and machine learning techniques.

Chapter 5: Mobile Network Traffic Prediction Framework: The ML-TP

This chapter proposes a meta-learning based mobile traffic prediction framework. The framework utilises the analysis technique proposed in Chapter 4. By exploiting the meta-

learning concept, a meta-learner is trained, giving the best initial parameters of the LSTM model according to the frequency-domain characteristics of a new mobile traffic prediction task.

Chapter 6: Mobile Network Traffic Prediction Framework: The dmTP

Based on the ML-TP in Chapter 5, an advanced mobile network traffic prediction framework is proposed to output both the optimal hyper-parameter value and initial status for the base-learner of a new base-task. In addition, the prediction performance of the proposed framework is investigated in terms of predicting the traffic load of different types of services and the mobile network traffic of different cities with different spatial scales. Furthermore, the performance in predicting varied mobile network traffic load is also tested.

Chapter 7:Conclusions and Future Work

This chapter summarises the works of this thesis and presents future work.

Chapter 2

Literature Review

2.1 Mobile Traffic Analysis

The network traffic refers to the amount of data passing through the network at a given time [90]. It is mostly encapsulated in network packets, which accounts for the load in the network. Network traffic is the main component for network traffic measurement, control, and management [90]. For mobile traffic, it refers to the traffic data generated in the cellular networks. Thus mobile traffic analysis denotes the analysis related to mobile traffic data.

The most crucial driving factor for mobile traffic analysis is that mobile traffic conveys vast and diverse individual information, such as geolocations, movement, interactions and mobile services accessed [1]. Meanwhile, mobile traffic data have three exceptional advantages. The first one is population coverage. Since more than 60% population around the world is equipped with at least one mobile device [3], mobile traffic data can represent human activities to a large extent. The second advantage is the geographical coverage. Mobile traffic data can be collected if a base station covers. Thus, the data can span from small geographical regions to city level, even nation-wide. The last one is time coverage. The timespan of mobile traffic data can range from weeks to months. Based on these advantages, mobile traffic analysis attracts more experts from various fields, such as sociology, epidemiology, transportation and wireless communications [1]. Consequently, research usually involves many domains, and it is difficult to find a clear separation for classification.

Since mobile traffic analysis is usually cross-disciplinary, a possible way for classification is to organise according to research subjects. From the perspective of telecommunications, based on the research's domains, the related research can be classified into three main domains: network analysis, mobility analysis, and social analysis [1]. The network analyses mainly focus on understanding the mobile network's traffic dynamics and discuss how the mobile network can accommodate these mobile network traffic demands; mobility analyses investigate either individual users or groups of users' mobility characteristics. The social analyses deal with the relationships between mobile traffic and social features, such as mobile users' interactions and how they use mobile services. Related research efforts will be discussed in the following sub-sections.

2.1.1 Mobile Network Traffic Prediction

In the past few decades, mobile cellular communication technology has developed rapidly. The system capacity and data rate have been improved significantly [91]. Along with recent advances in mobile communication technologies, the number of mobile devices experienced explosive growth, including a wide variety of smart devices, such as the Internet of things devices. According to the investigation from GSMA [3], the number of mobile subscribers has exceeded five billion in 2020. The rapid growth of mobile devices and the demand for multimedia services prompt the rise of data traffic. The global mobile data traffic has an eighteen-fold growth from 2012 to 2016. It is estimated to increase seven-fold in the following five years [63]. The global mobile data traffic volume has reached 19.01 exabytes (10^{18} bytes) per month and will increase at a 46% annual growth rate [64]. The industry's latest prediction indicates that the annual traffic generated in 2021 will reach 3.3 zettabytes (10^{21} bytes) [92].

Although mobile network operators and network equipment vendors keep making efforts on enhancing wireless link bandwidth and network capacity by employing advanced techniques at both medium access layer and physical layer in Long Term Evolution (LTE) and LTE-Advanced systems, it is difficult to sustain the rapid growth of network demand [91]. The advances in hardware are still difficult to meet the rapidly increasing demand, so re-

searchers turned to seek improvements at the software level to improve network performance [91].

To meet the rapidly growing traffic needs and reduce the OpEx, mobile network traffic prediction at the cellular level plays a vital role in mobile networks since many mobile applications rely on real-time or approximately real-time traffic analysis throughout a considered RAN [66]. The reason behind this is that, by forecasting the traffic in mobile networks, some proactive optimisation strategies can be applied to improve network performance. For example, the sleeping strategy for the base stations might be carried out to reduce energy consumption, which accounts for a significant proportion of OpEx [37]. By knowing the network traffic in advance, a resource allocation strategy can be applied, avoiding the possible network congestion [69]. Also, for mobile caching, a key element in 5G networks whose purpose is to proactively store contents in edge clouds adjacent to mobile users in advance in order to subscribers' experience and the stability of mobile networks [93], also requires predicting downlink mobile traffic data required by terminal devices. Moreover, with the introduction of network virtualisation and artificial intelligence, future mobile networks will operate in an ultra-flexible way [70, 71]. Thus mobile network traffic prediction becomes an essential element in network design [94]. Therefore, mobile traffic prediction will then be more and more critical to mobile networks, including network deployment, management, and even acquiring intelligence [61, 73].

Since analysing mobile network traffic is the key to know how the resources in the mobile network are consumed and even yield the potential of improving mobile network performance, it attracts more attention. The early research focuses on exploring mobile network traffic characteristics, which lays the foundations for subsequent research. Williamson et al. [75] observed the network traffic of 100 base stations. They found that network traffic patterns show periodicity on a daily basis. Tidal effects occur where high traffic consumption during the day and low consumption at night. Paul et al. [12] conducted the auto-correlation analysis of the whole network traffic in a nationwide 3G network. They showed that the auto-correlation function has peaks at 24-hour intervals, indicating the regularity of human activity patterns. Shafiq et al. [76] presented an analysis of aggregated cellular network

traffic from a mobile operator. They showed that the aggregated traffic load has diurnality. The regularity is also confirmed by Keralapura et al. [95], Zhang and Arvidsson [96], and Oliveira et al. [97]. Although the periodicity and tidal effects exist, mobile network traffic also shows fluctuations over the daily patterns and geographical locations. This diversity also depends on the dataset considered [75, 98]. At the same time, the mobile network traffic is also spatially heterogeneous. Hoteit et al. [99] analysed the service consumption of mobile networks in Paris and showed that mobile traffic differs geographically. Shafiq et al. [100] noticed that the types of mobile traffic consumed depend on the locations. Therefore, the regularities of mobile network traffic make predicting mobile network traffic feasible and fluctuations make the prediction a challenge.

Many efforts have been made to use mathematical models to model mobile network traffic. Jin et al. [101] characterised the data usage patterns by the Markov model. Shafiq et al. [76] employed a Markov model to model the cellular network's temporal traffic demand. However, due to the Markov model's limitation, which has limited states, it can only model limited and discrete traffic status. In contrast, the nature of real network traffic is continuous. In light of mobile network traffic's self-similarity, Ge et al. [15] utilised the α-stable model to predict the cellular level traffic. Tikunov and Nishimura [78] proposed a cellular traffic prediction technique to forecast the traffic in GSM/GPRS networks using Holt-Winter's exponential smoothing model. Wang et al. [89] analysed the temporal cellular traffic characteristics and showed that it has periodicity. They further found the mobile traffic has three main frequency components. Then they proposed a sinusoid superposition model to describe the temporal traffic dynamics. The linear AutoRegressive Integrated Moving Average (ARIMA) model, which is a generalisation of an autoregressive moving average model, aiming to address the time-series modelling and prediction, was adopted in [41, 79, 102, 103] to capture the short-term correlations in mobile network traffic. As an extension of the ARIMA method, the seasonal ARIMA model has been adopted in [86, 103] to improve the ARIMA model on long-term traffic correlation capturing. Although these methods have good understandability and relatively low computational complexity, it is difficult for them to achieve good prediction

accuracy since mobile network traffic is much more complicated, where mathematical models are hard to model mobile network traffic irregularities.

2.1.2 Mobility Prediction in Mobile Networks

Previous research has shown that user mobility affects mobile network research and development [104], such as mobility management for user handover and association events in which a user moves between cells [105]. To manage user mobility and keep connections between users and the mobile network, mobility prediction is considered a practical approach [106]. The ability to predict the subscriber's next cell or even the trace that the subscriber will move across cells becomes a critical aspect in the future mobile networks [107]. Mobility prediction has three potential applications in mobile communications: handover management, resource management, location-based service pre-configuration, and network planning [108].

Early research focuses on analysing user mobility characteristics. Tang and Baker [109] investigated the user movement features in a metropolitan-area packet radio wireless network. They used the clustering method and focus on three aspects: the frequency, distance of moving, and potential mobility patterns. The results showed that most users have very limited mobility. The travel distance is inversely correlated to the number of locations visited by a user. They also found that different users show different mobility patterns, such as some users are active all day. In contrast, some are more active only during the daytime. Halepovic and Williamson [13] also obtained similar results. They found that the majority of users have low mobility, and the mobility follows a heavy-tailed distribution, meaning that very few users have high mobility. Paul et al. [12] also proved this heavy-tailed distribution of visited locations with a national-wide dataset. The subsequent works [24, 26] also obtained similar results.

Later, research efforts have been made to explore the spatiotemporal regularity of user mobility. González et al. [27] have shown that the popularity of locations visited by a user follows Zipf's distribution, which means users are more likely to visit specific locations more frequently. Besides, they showed that a strong regularity exists in individuals' mobility patterns, i.e., users are likely to re-visit some locations within 24 hours, indicating the

temporal regularity. Song et al. [28] conducted a similar analysis using their dataset to prove Zipf's law and mobility regularity on a 24-hour basis. Cho et al. [29] and Hess et al. [26] also showed that user mobility has geographic and temporal regularity, i.e., users tend to visit the same locations at a similar time every day. The latter further indicated that besides daily pattern, user mobility also shows the periodicity of one week. This is consistent with the results in [30] and [31]. These strong regularities raise the discussion on the predictability of user movement. Song et al. [32] conduct an early investigation on the predictability of user mobility. They employed the entropy method to measure the randomness and measured the regularity of the spatiotemporal sequence of locations that a user visits. Their dataset has 50,000 users' trajectory records, and the results show that users' movements have low randomness. 93% of user mobility is theoretically predictable. This research lays the foundation for predicting the feasibility of user mobility. Later research also showed that high predictability exists in user movements [14].

Many efforts have been made to model user mobility from both individuals and aggregated population perspectives. The individual perspective mainly focuses on analysing and predicting individual user moving traces. In contrast, the aggregated mobility analysis aims to investigate a large population's mobility features with low spatial granularity [1]. Scourias and Kunz [83] proposed a stochastic mobility model based on daily activity patterns for individual mobility studies. In this model, the subscribers were divided into four categories depending on their jobs and activities and then derived a mobility model based on activity pattern theory. In [13], Halepovic and Williamson first obtained the empirical distribution of the number of cells visited by a user and the empirical distribution of cell change frequency and then built a stochastic-based mobility model.

The Markov chain model is one of the popular methods used in mobility prediction. It is a stochastic process that describes a sequence of all possible states. The probability of each state only depends on the previous state [16]. A Markov process is memoryless, meaning that the future state is solely based on the present state. Thus it is usually used in scenarios with a chain of events that the next event is only dependent on the current status and independent of the previous status. The transition probability matrix is pre-defined. Due

to this property, the Markov chain has become a popular tool in mobility prediction. Ulvan et al. [17] and Ariffin et al. [18] employed the original Markov chain model to predict user mobility. A user's positions may visit from the states, and the user moves among states with transition probability. The next state that the user may visit only depends on the current state. In [19], the state is not limited to a position but can also represent the movement. Hadachi et al. [20] proposed an enhanced Markov Chain model that embeds association rules, such as universal users mobility behaviour and temporal rule, into a second-order Markov Chain model. However, in reality, the next user state depends not only on the current state but also on previous states. Gambs et al. [89] proposed a mobility Markov chain model, which considers n previous locations that a user has visited to solve this issue. By incorporating more previous locations, the prediction accuracy is improved. However, this improvement comes at the cost of increasing the model complexity. Markov chain model has two disadvantages. Since its performance depends on the transition probability matrix, how to obtain the accurate matrix is a challenge. The other disadvantage is the poor extensibility. When the number of states is large, acquiring the transition matrix becomes challenging.

For the efforts on aggregate mobility analysis, the Gravity model is used in [22] to model user mobility. It performs well in modelling the commuting distances of users. Simini et al. [23] proposed the radiation model, which outperforms the gravity model. Lu et al. [14] employed the Markov chain model. They showed that the first-order Markov chain model could achieve an average prediction accuracy of 91%. Since the gravity model and the Markov chain model can only model the mobility at low spatial granularity, Yang et al. [25] propose an improved model by combining these two models. The results show that the new model works at different granularities.

This thesis mainly focuses on individual mobility since mobility on the individual level is of more interest in cellular networks in most cases.

2.1.3 Social Analysis in Mobile Networks

The social analysis focuses on the relationship between mobile traffic and user social characteristics, such as investigating the impacts of demographic, economic, or environmental

factors on mobile service consumption [1]. The social analysis can be classified into four main research directions [1], as illustrated as follows.

The first research direction aims to understand the structure of interactions among mobile users. Most studies represent these interactions through graph theory [1]. The mobile traffic datasets are usually represented as mobile call graphs. A mobile call graph can be regarded as a mathematical structure recording subscribers and mapping to a set of vertices with their interactions, such as voice calls and text messages [1]. In an early work [2], Nanavati et al. described the call-in and call-out relationship by constructing an unweighted directed graph. They found that the call-in and out behaviours follow a power-law distribution, which implies that users who make calls to more users also tend to be called by more subscribers. Doran et al. [5] have drawn a similar conclusion.

The second direction is demographics. This direction studies the relationship between mobile user behaviour and demographics. For example, Yang et al. [6] investigated the impacts of age and gender on mobile traffic. They found that people of a similar age tend to communicate more often. Besides, female users have a longer calling duration than male users. Soto et al. [7] defined a list of mobile user features and showed that the mobile features could predict the user's economic level.

The impacts of the geographical and social environment on mobile communication patterns are the third research direction. Onnela et al. [110] investigated the effects of physical distance on social interactions. They indicated that the mobile contact between two users follows a power-law distribution with respect to their geographical distance. In [111] and [112], the authors implied that the users who have close social contact tend to reside within a short geographical distance.

The last direction is the relationship with epidemics. Wesolowski et al. [8], Enns and Amuasi [9] studied the correlation between mobile traffic and diseases diffusion. Besides using mobile traffic to understand the disease spreading, some research also tries to utilise mobile traffic to control the disease spreading. Leidig et al. [10] and Kafsi et al. [11] proposed spreading-aware strategies to reduce disease diffusion by extracting trajectories from mobile traffic.

2.2 Machine Learning in Mobile Network Traffic

With the rise of machine learning, lots of efforts have been made to solve the problems as mentioned above using machine learning techniques. For mobile network traffic prediction, at the early stage, Linear Regression (LR) [80], compressive sensing [81, 82, 113, 114], and the Support Vector Regression (SVR) [115] are utilised for traffic loads prediction. Researchers [116–119] also used principal components analysis [116], Kalman filtering [116-117], and Gaussian process [119] to address the traffic prediction tasks. These machine learning methods are shallow machine learning approaches due to the low complexity of these algorithms. The low complexity comes at the cost of limited prediction performance. As the mature of deep learning algorithms, some research efforts leveraged deep learning algorithms to predict mobile network traffic.

Nie et al. [120] exploited the deep belief network-based model and the Gaussian model to model the low-pass and high-pass features of cellular level mobile traffic. Wang et al. [121] investigated the spatio-temporal dependencies among cellular towers and used graph neural networks to model and forecast mobile traffic. Considering the capability of capturing the temporal correlations, Tian and Pan [84] trained an RNN to predict traffic loads. As an evolution method of RNN, LSTM shows competitive performance in capturing long-term mobile traffic dynamics and thus has been used in many related works, such as [122–124]. Zhang et al. [125] considered the mobile traffic pattern as images. They proposed a ZipNet, which combines a convolutional neural network and a generative adversarial neural network to capture spatio-temporal mobile traffic patterns. Huang et al. [126] combined the convolutional neural network and RNN, where a convolutional neural network captures the geographical features, and RNN extracts mobile traffic's temporal features. However, although these deep learning-based methods improve traffic prediction accuracy, they incur some new challenges. First, deep learning-based techniques require lots of data to train the model, meaning they have high data availability requirements. Secondly, these existing methods must construct and train a specific prediction model for each individual prediction task since the time series of mobile traffic generated in different cells are quite different.

Separately training the prediction models for different mobile cells is not only time consuming but also computing consuming.

For mobility analysis, machine learning is becoming a powerful tool due to its capability of capturing spatial dependencies from sequential data [47]. Biesterfeld [83] made early attempts to use Artificial Neural Network (ANN) to learn the subscriber movement pattern and predict the next location during the next time interval. Their results show that ANN outperforms conventional methods in dealing with dynamic movement patterns. Akoush et al. [127] proposed a Bayesian neural network, which integrates Bayesian inference into ANN to predict the next location and service required by the user. Chen et al. [128] proposed a prediction framework that uses an echo state network with conceptors, which is a particular type of RNN, to predict user mobility patterns and content request distributions. Yang et al. [129] adopted a support vector machine for terminal mobility prediction in 5G ultra-dense networks, which reduces the hardware complexity compared with ANN but can still achieve competitive prediction accuracy. In [130–132], principle component analysis was employed as another approach to extract user mobility patterns. Xi et al. [133] argued that mobility data are high-dimensional as the development of mobile networks. The shallow machine learning models, such as support vector machine, have difficulties dealing with such high-dimensional data. Nguyen [134] made early efforts to use a deep learning algorithm, deep autoencoder, a variant of ANN that stacks multiple layers of restricted Boltzmann machines to learn typical user mobility patterns. The results show that the deep autoencoder framework significantly improves the performance in reconstructing user trajectory than principal components analysis. In [133], Xi et al. employed a convolutional neural network, which can capture the local dependency of visual information. They built a convolutional neural network model with a hierarchical structure to predict human moving paths. However, although a convolutional neural network achieves excellent performance, it is hard to deal with the long time-series data.

Chapter 3

Fundamentals of LSTM

This chapter mainly focuses on introducing the fundamentals of the LSTM network. The principles of the LSTM network and how it is used for prediction are introduced in detail. Besides, the data pre-processing techniques are also presented along with the employment of LSTM in this chapter. By predicting mobile user mobility, the LSTM network shows great potential in dealing with time series problems in mobile networks.

3.1 Introduction

The traditional ANN is inspired by biological neural networks such as human brains. Figure 3.1 shows the architecture of a simple ANN example. It has two neurons in the input layer and output layer, one hidden layer which contains three neurons. Each neuron takes one value, a linear combination of all input connections, as the input and outputs one value. In ANN, all inputs are independent of each other, and all outputs are independent as well. This means that the output is fixed for a given input vector, and changing input orders will not change the output values. For example, for two given input samples, $\mathbf{x}_1$ and $\mathbf{x}_2$, the ANN gives the output denoting as $\mathbf{y}_1$ and $\mathbf{y}_2$, respectively. Changing the input order to $\mathbf{x}_2$, $\mathbf{x}_1$ will not change the results given by ANN except the output order, i.e., the output becomes $\mathbf{y}_2$ and $\mathbf{y}_1$. However, in practice, the inputs and outputs are correlated in some tasks. For example, the user movement in the mobile network is temporal correlated. The next location that the

user will move to depends on the current location [135]. Hence, to address this correlation, RNN, a type of ANN, is utilised. Figure 3.2 shows the simplified network architecture of ANN and RNN, respectively. The difference between ANN and RNN is that RNN has a loop in the hidden layer, which allows the information to persist. By adding the feedback connections, the information from previous inputs can keep passing such that previous states have an influence on future states, as shown in the unfolded RNN architecture in Figure 3.2. This enables RNN the ability to deal with sequential tasks. However, RNN has the problem of vanishing gradient [136]. This means when training an RNN using the back-propagation technique, and the back-propagated gradients will tend to zero, i.e., vanish, leading to the network loses previous information [137]. This leads to the limitation that RNN cannot deal with tasks with long-term dependencies. In practice, some tasks, such as natural language processing and time-series prediction tasks, are sequentially related, which means previous inputs will affect subsequent outputs. In these tasks, long-term dependencies are essential factors. To solve this problem, LSTM, an improved architecture of RNN, was proposed [138]. The key idea of LSTM is to replace the loop structure in RNN with gate architecture known as the memory block [138], which determines what and to what extent that the information will be remembered. The gate structure significantly mitigates the gradient issues in traditional RNNs. This elegant feature makes LSTM a powerful tool in dealing with sequential tasks, such as speech recognition [139], handwriting recognition [140], and anomaly detection [141].

The critical component that makes LSTM networks possess the ability to model long-term dependencies is the LSTM memory block. As illustrated in Figure 3.3, each LSTM memory block is a recurrently connected subnet logically, which contains some functional modules called gates.

According to their corresponding practical functionalities, these gates are classified as the input gate $\mathbf{i}_t$, forget gate $\mathbf{f}_t$ and output gate $\mathbf{o}_t$. In addition, LSTM has a cell state $\mathbf{C}_t$ which stores previous network state information from previous inputs. The functions of the gate structures are summarised as follows:

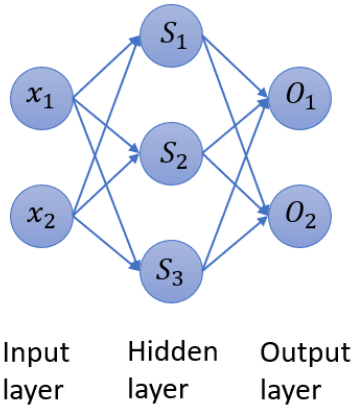


Fig. 3.1 An example of ANN

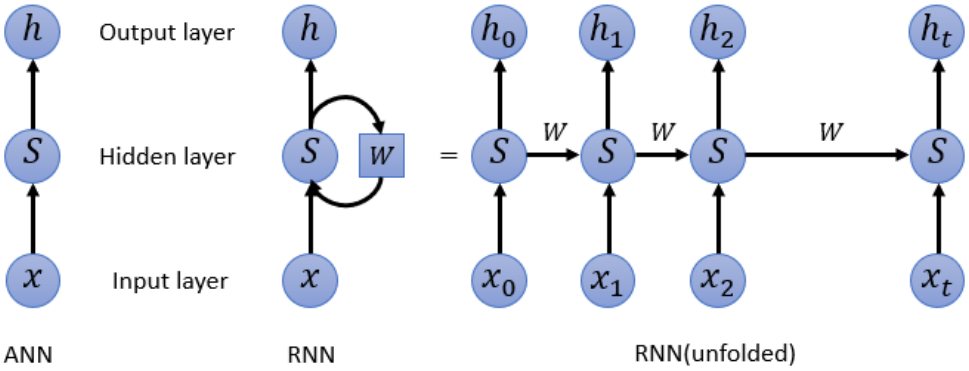


Fig. 3.2 Network architecture of simplified ANN (left) and simplified RNN (right)

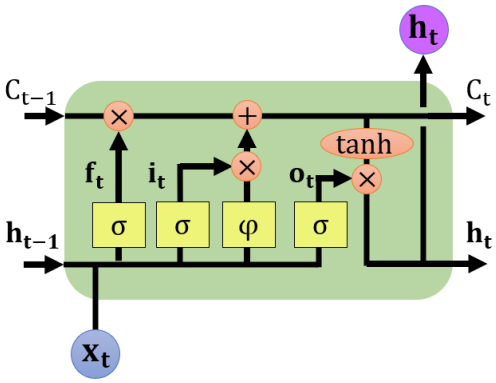


Fig. 3.3 Inner structure of an LSTM network

(i) The forget gate controls how much information still remains in the memory block's current state through the recurrent connection (see equation 3.1).

(ii) The input gate controls how much new information flows into the memory block's current state (see equation 3.2).

(iii) Both the forget gate and the input gate control the cell state, which is represented in equation 3.3.

(iv) The output gate controls how much information is used to compute the output activation of the memory block and further flows into the rest of the LSTM network (see equation 3.4).

The main operations conducted in the LSTM network are concluded as below:

$$\mathbf{f}_t = \sigma(\mathbf{W}_{\mathrm{xf}}\mathbf{x}_t + \mathbf{W}_{\mathrm{hf}}\mathbf{h}_{t-1} + \mathbf{b}_{\mathrm{f}}) \tag{3.1}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_{\mathrm{xi}}\mathbf{x}_t + \mathbf{W}_{\mathrm{hi}}\mathbf{h}_{t-1} + \mathbf{b}_{\mathrm{i}}) \tag{3.2}$$

$$\mathbf{C}_t = \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \varphi(\mathbf{W}_{\mathrm{xc}}\mathbf{x}_t + \mathbf{W}_{\mathrm{hc}}\mathbf{h}_{t-1} + \mathbf{b}_{\mathrm{c}}) \tag{3.3}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{\mathrm{xo}}\mathbf{x}_t + \mathbf{W}_{\mathrm{ho}}\mathbf{h}_{t-1} + \mathbf{b}_{\mathrm{o}}) \tag{3.4}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t) \tag{3.5}$$

where $\odot$ denotes the Hadamard product, $\mathbf{x}_t$ is the input vector, $\mathbf{h}_t$ are the hidden states, $\mathbf{W}_{\mathrm{xf}}$, $\mathbf{W}_{\mathrm{hf}}$, $\mathbf{W}_{\mathrm{cf}}$, $\mathbf{W}_{\mathrm{xi}}$, $\mathbf{W}_{\mathrm{hi}}$, $\mathbf{W}_{\mathrm{ci}}$, $\mathbf{W}_{\mathrm{xc}}$, $\mathbf{W}_{\mathrm{hc}}$, $\mathbf{W}_{\mathrm{xo}}$, $\mathbf{W}_{\mathrm{ho}}$ and $\mathbf{W}_{\mathrm{co}}$ represent the weight matrices in each gate that are needed to be trained. $\mathbf{b}_{\mathrm{f}}$, $\mathbf{b}_{\mathrm{i}}$, $\mathbf{b}_{\mathrm{c}}$ and $\mathbf{b}_{\mathrm{o}}$ are the biases, which are 'offsets' added to each unit to ensure the activation function to be shifted to fit the data better. $\sigma(x)$ and $\varphi(x)$ are activation functions where $\sigma(x)$ usually takes the sigmoid function and $\varphi(x)$ usually takes the hyperbolic tangent function. The reasons for choosing these two functions are that the output of the sigmoid function ranges between 0 and 1, which can either let no flow (0) or complete flow (1) of information go through the gates. To overcome the vanishing gradient problem, a function whose second derivative can sustain for a long time before

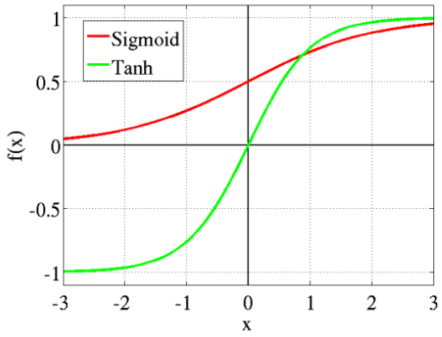


Fig. 3.4 Sigmoid function and hyperbolic tangent function

going to zero is needed. Thus the hyperbolic tangent function is a good candidate with this property. In addition, it usually converges faster and consumes less gradient computation [142]. The sigmoid function of the hyperbolic tangent function is defined as follows:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{3.6}$$

$$\varphi(x) = \tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{3.7}$$

Figure 3.4 shows the sigmoid function and hyperbolic tangent function, which are usually employed in the LSTM network. In a basic LSTM memory block, each gate consists of a two-layer neural network where the neuron number of the output layer equals the output vector's length. The neuron number of the input layer equals the input vector's length plus that of the output vector. In Figure 3.3, $\oplus$ and $\otimes$ denote summation and dot product of two vectors, respectively.

The training process of a traditional ANN (feedforward neural network) is based on gradient descent combined with a backpropagation algorithm. The method used to train an LSTM is slightly different. Since LSTMs and RNNs have cycles, the backpropagation algorithm cannot directly be applied. The backpropagation through time algorithm is proposed [143] for training LSTMs and RNNs. It works similarly to the backpropagation algorithm. It first unfolds the LSTM in time, transforms LSTM into a feedforward neural network, and then applying the backpropagation algorithm.

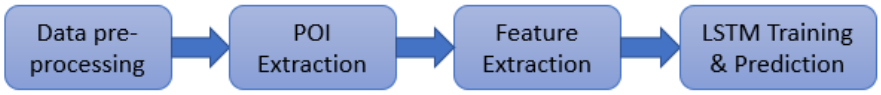


Fig. 3.5 General process for LSTM mobility prediction

3.2 LSTM in Mobile Communications

Mobile networks generate various sequential data, such as network traffic loads and user equipment trajectories [47]. Thus, methods dealing with the time series problems, such as LSTM, have become a promising approach to enhancing sequential task analysis. This section will introduce the general process and background knowledge of using LSTM to predict user mobility. The overall steps for mobility prediction using LSTM in this chapter have been summarised in Figure 3.5. It consists of four steps. First of all, the original data, such as GPS data, are needed to be pre-processed to ensure the data are suitable for this task; secondly, the user Points of Interest (POI) will be extracted; the POI represents the regions where user spends long time to stay at, and it will be introduced in detail in the following part. Then the features for training the LSTM network will be extracted. Finally, an LSTM network will be trained and used for mobility prediction. The rest of this section will introduce the procedure step-by-step.

3.2.1 Data Pre-processing

Data collection and pre-processing are two critical parts that can affect the performance of prediction performance. For a trajectory recorded by GPS logs, although GPS usually has relatively high accuracy in recording geolocation data, it may still have errors due to some reasons such as noise and time error. Therefore, to reduce the effects of data error on the prediction performance, data pre-processing is necessary.

A GPS trajectory usually contains a set of GPS records. Each record has timestamp and geolocation information, including latitude, longitude, and altitude. It can be denoted as

$$p_{(i)} = (\text{lat}_i, \text{lon}_i, \text{alt}_i, t_i) \tag{3.8}$$

where i represents the i-th GPS record, lat_i, lon_i, alt_i, t_i, represent latitude, longitude, altitude, and time, respectively.

To identify the error GPS records, a heuristic anomaly detection method is applied, which has been widely used in previous work [144–146]. The mean speed between the GPS point and its preceding point can be calculated, denoted as

$$v_{p_{(i)},p_{(i-1)}} = \frac{\text{dist}\left(p_{(i)}, p_{(i-1)}\right)}{t_i - t_{i-1}} \tag{3.9}$$

where the denominator and the numerator represent the geographical distance and time difference between the i-th record and its preceding record, i.e. the $(i-1)$-th record, respectively.

The speed obtained is then compared with a pre-set threshold speed v_{th}, which depends on the maximum reasonable speed in practice, such as urban speed limit and maximum achievable speed of pedestrians, vehicles, or railways. If the speed exceeds the threshold speed, i.e., $v_{p(i),p(i-1)} > v_{th}$, the i-th record will be regarded as an anomaly. In this work, the record will be removed if it is identified as an abnormality.

After the error detection, data resampling will be applied. The motivation behind this is that the GPS sampling frequency is usually not uniform. The time intervals between consecutive neighbouring records range from seconds to hours. Since this work focuses on the locations where the user stays for a long time rather than the trajectory itself, and the nonuniformity makes the latter feature extraction becomes difficult. Thus, the resampling technique is needed.

As mentioned above, this work concentrates on the areas where a user spends a long time rather than precise location information. Thus the average coordinates (latitude and longitude) can be taken during each time interval as the rough location. The average latitude and longitude during the time interval j can be defined as:

$$\overline{lat}_j = \frac{1}{m}\sum_{k=1}^{m} \text{lat}_j, \forall k \text{ in time interval } j \tag{3.10}$$

$$\overline{lon}_j = \frac{1}{m}\sum_{k=1}^{m} \text{lon}_j, \forall k \text{ in time interval } j \tag{3.11}$$

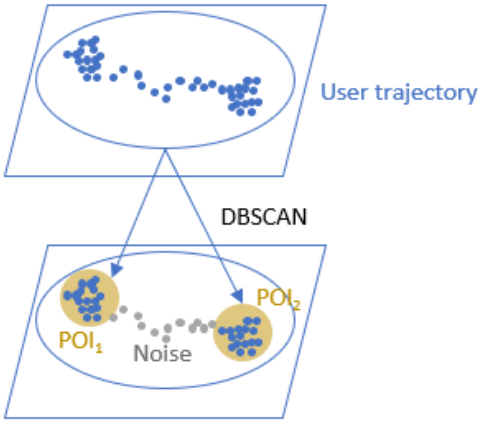


Fig. 3.6 User trajectory and POI extraction

where m is the total number of samples in time interval j.

Thus, the average coordinate at time interval j can be denoted as coordinate$(i) = (\overline{\text{lat}_i}, \overline{\text{lon}_i})$

3.2.2 POI Extraction

After the pre-processing, a trajectory with a fixed sampling rate is obtained. This trajectory describes the user mobility characteristics. Not all points in the trajectory are equally important. This work focuses on the locations where users spend time, such as shopping centres, working areas, and residential areas. For example, Figure 3.6 shows an example of a user trajectory that is sampled at a fixed frequency. There are two areas with densely distributed GPS points through observation. These GPS densely distributed areas represent the locations where the user stays for a long time. These areas are named POI, and these POIs are of more interest. To extract these locations, the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) clustering method is applied as follows.

DBSCAN and K-means are the most common clustering approaches. The reasons for choosing DBSCAN in this work can be summarised as follows. Compared to the very traditional K-means method, the DBSCAN based clustering approach does not need to specify the number of clusters. Since the POI amount of each user is unknown, the K-means

approach is not suitable for this case. Second, DBSCAN is more robust to noise than K-means. Since POIs are of more interest than the moving path in this work, DBSCAN can recognise the clusters and mark unrelated points as noise. Compared with DBSCAN, K-means will assign every point into a cluster. Thus it is more sensitive to noise. In addition, DBSCAN can deal with the irregular shapes of clusters [147]. As shown in Figure 3.6, the extracted POIs are of more interest, and the points forming the moving path are regarded as noise. When using K-means, these grey points will also be classified into clusters. Compared with K-means, DBSCAN can effectively recognise the noise points and filter them out. Thus, DBSCAN is a better choice for POI extraction.

In general, there are two parameters to be specified in the DBSCAN algorithm, known as the Eps and the MinPts. The effects of the two parameters on the clustering results have been concluded in **Remark 3-1**. In this study, Eps represents the physical distance, and MinPts is a threshold to form a POI. A point is considered as the core point of other data points if the number of these data points within the radius of Eps is no less than MinPts. The purpose of clustering is to find the set of all points connected to the cluster's core point. The clusters obtained via the DBSCAN technique are POIs of the user. The procedure of DBSCAN can be summarised as **Algorithm 3-1**, as shown below.

Remark 3-1: The value of Eps affects the shape of clusters. Large Eps may introduce the unrelated points into clusters, and small values may lead to one region be divided into several clusters. MinPts determines the minimum number of points to form a cluster. The value selection of Eps and MinPts is based on experience.

By applying the DBSCAN algorithm to the pre-processed user trajectory, a set of clusters will be obtained. These clusters represent the areas where the user stays for a long time, i.e., each cluster denotes a POI. Then each POI will be numbered, and all POIs form the POI set of the user.

3.2.3 LSTM Network Training and Prediction

This section illustrates the related work for training an LSTM network, including input feature selection, feature normalisation, output selection, dataset division, and model performance

Algorithm 1 : POI Extraction using DBSCAN

Input: P: Pre-processed trajectory database consisting P records;
Eps: radius of distance;
MinPts: minimum number of points required to form a cluster (POI);
Output: Collection of density-based clusters

```
Mark all objects p as unvisited;
C = 0;
for i=0, i<length of P, i++ do
    if p[i] is unvisited then
        Mark p[i] as visited;
        Find the set N which includes all objects in the range of Eps;
        if |N| ≥ MinPts then
            C++;
            for p[j] in N do
                if p[i] is unvisited then
                    Mark p[i] as visited;
                    Find the set N' which includes all objects in the range of Eps;
                    if |N'| ≥ MinPts then
                        Add all points in N' into N;
                    end if
                    if p[j] does not belong to any cluster then
                        Add p[j] to cluster C;
                    end if
                end if
            end for
        end if
      return C;
    else
        Mark p[i] as noise;
    end if
end for
```

evaluation. These processes can be summarised in Figure 3.7. In this figure, the general procedure for mobility prediction and evaluation can be divided into five parts: network input feature normalisation, LSTM training, LSTM prediction, inverse normalisation, and prediction evaluation. In this figure, $\mathbf{x}_t$ is a vector which contains the features of the user's historical trajectories, $\mathbf{x}'_t$ is the normalised input vector to the LSTM network; $\mathbf{o}_t$ is the output given by the LSTM network, which further gives $\widehat{c}_{t+1}$ after inverse normalisation. Finally, $\widehat{c}_{t+1}$ is compared with ground-truth, c_{t+1}, to evaluate the model performance.

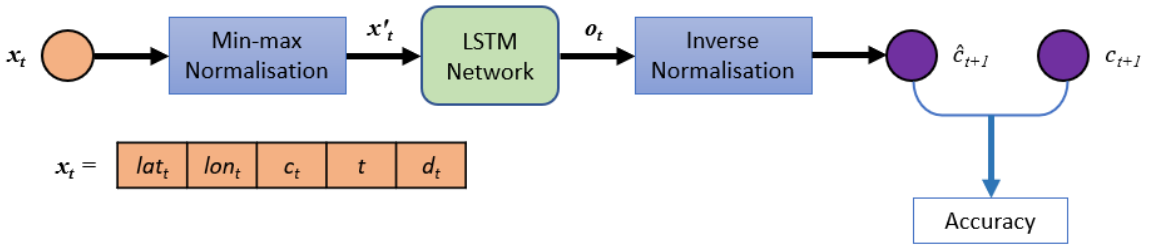


Fig. 3.7 General procedure for training an LSTM network for mobility prediction and evaluation

In this work, the LSTM aims to predict the POI that the user will go to during the next time interval. Therefore, the output of the LSTM network will be the label of the POI. The LSTM network input is a vector consisting of five features: latitude, longitude, POI, date, and time. Thus, each data sample will be converted to the form which is suitable to the input of the LSTM, i.e., each sample is denoted in the following form:

$$\mathbf{x}_t : (\mathrm{lat}_t, \mathrm{lon}_t, \mathrm{c}_t, d_t, t) \tag{3.12}$$

where lat_t, lon_t, c_t, d_t, and t represent latitude, longitude, POI label, day of the week and time of sample $\mathbf{x}_t$, respectively.

For each feature, a conversion is needed to ensure the feature is compatible with the LSTM network. To be more specific, the latitude and longitude are in floats, which can be input to the LSTM network; however, the date ranges from Monday to Sunday, the value 'Monday' cannot be directly inputted to the LSTM network. An encoding scheme is necessary to convert the non-digital features into digital form. Since the date has a very limited value range, i.e., only seven possibilities, the simplest way is to encode Monday to Sunday with integers 1 to 7, respectively. Similarly, for the feature time, '0, 1,..., 23' is used if it is in hours, or '0, 1, ..., 1439' is used if it is in minutes.

After feature encoding, feature normalisation is applied to ensure the LSTM network work properly. Min-max normalisation is one of the most common methods for feature scaling. It scales features to the range in [0,1] or [-1,1]. The former feature range is more

common, and it is defined as:

$$y_{\text{norm}} = \frac{y - \min(y)}{\max(y) - \min(y)} \tag{3.13}$$

where $\min(y)$ and $\max(y)$ represent the minimum and the maximum value of feature y.

The min-max normalisation applies to all features, respectively, to ensure all features of the samples are in the range [0,1]. There are two more steps before training the LSTM network. The first one is to clarify the output of the LSTM network, i.e., the target. In this work, as mentioned at the beginning of this section, the target is the POI that the user will go to during the next time interval, i.e. c_{t+1}. Therefore, each sample has an input-output pair, denoting as $(\mathbf{x}_t, c_{t+1})$. Inverse normalisation applies to the output of the LSTM network, o_t, to obtain the predicted POI given by the network, denoted as $\widehat{c}_{t+1}$. The calculation of inverse normalisation is as follows

$$y_{\text{inv}} = y_{\text{norm}} \left(\max(y) - \min(y)\right) + \min(y) \tag{3.14}$$

where y_{norm} is the normalised variable, $\min(y)$ and $\max(y)$ are the same as in equation 3.13.

The final step is to divide the dataset into a training dataset and a test dataset. The training dataset is used to train the LSTM network, and the test dataset is used to evaluate the performance of the LSTM network. The training dataset usually accounts for 70-80% of the whole dataset, whereas the remaining is the test dataset.

The architecture of the LSTM network employed in this chapter consists of three layers: one input layer, one hidden layer and one output layer. The input layer has ten neurons that take the input feature as the network input; the hidden layer has five neurons. The output layer has one neuron that outputs the normalised predicted POI. The LSTM network is trained based on a widely used stochastic gradient-based optimisation technique, Adam [148]. In addition, the MSE loss is chosen as the loss function.

To evaluate the performance of the LSTM network, prediction accuracy is employed, which measure the number of correct predictions over the total number of predictions. The

accuracy can be defined as:

$$\text{Accuracy} = \frac{1}{m}\sum_{i=1}^{m} 1_{\hat{c}_i=c_i} \times 100\% \tag{3.15}$$

where c_i is the ground truth of the POI in the time interval i, and $1_{\hat{c}_i=c_i}$ represents value is counted as 1 when $\hat{c}_i = c_i$.

3.3 Use Case: The LSTM-based User Mobility Prediction

In this section, a user mobility prediction model is proposed using the LSTM network. The data pre-processing and user movement feature extraction are introduced first. Then the LSTM network is utilised to predict user mobility. The model is validated using a real dataset.

3.3.1 Data Pre-processing

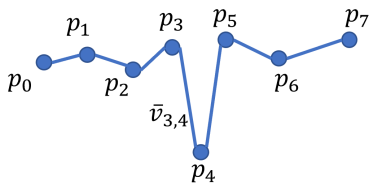


Fig. 3.8 An example of user trajectory

The dataset used is collected by the Geolife project from Microsoft Research Asia [149]. It has 182 users' trajectories in a period of three years. Each record contains GPS location, date, and time, which can be expressed as $p : (\text{lat}_p, \text{lon}_p, \text{alt}_p, t_i)$, representing latitude, longitude, altitude, date and time, respectively. The altitude information is ignored in this work since this work focuses on the horizontal position. The records are collected under different time intervals, from seconds to hours. Figure 3.9 shows the original trajectory of a user in a day. By zooming in the trajectory, it can be found that the trajectory is composed of many dense GPS points. Each GPS point is a record, and some records may have the wrong location due to the error of the GPS. The method introduced in section 3.2.1 is used to detect and remove

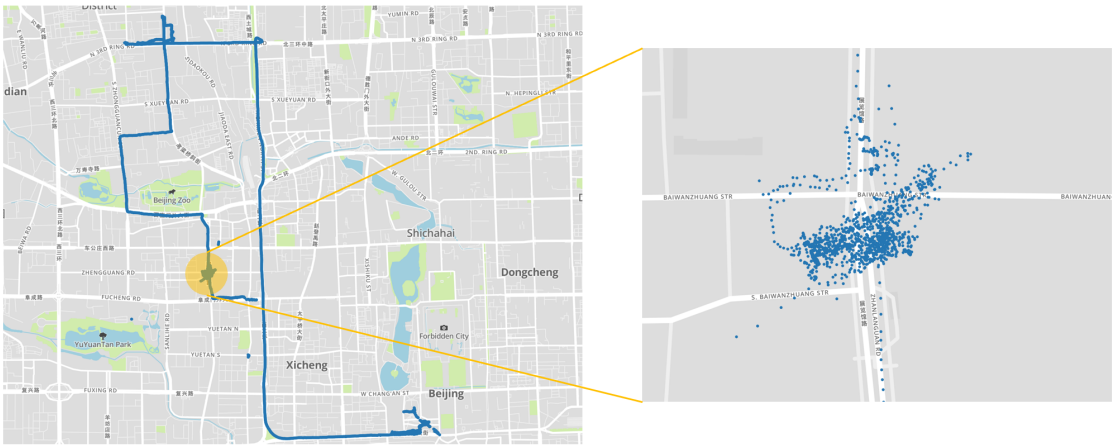


Fig. 3.9 Original trajectory of a user

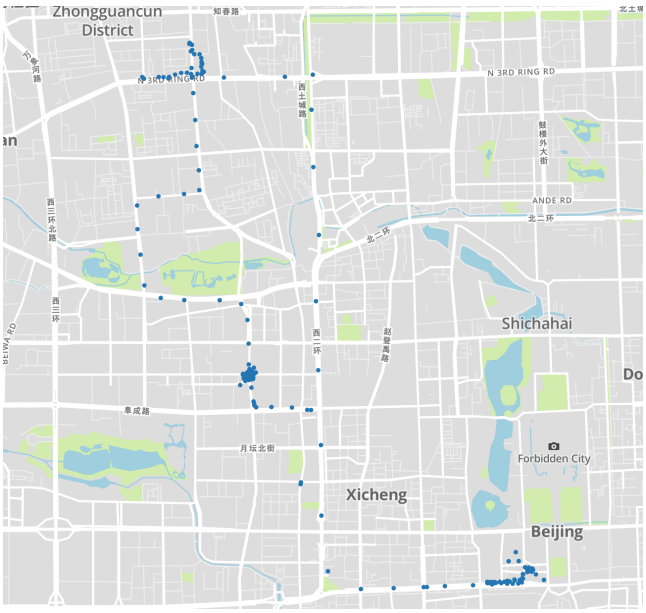


Fig. 3.10 Pre-processed user trajectory

these abnormalities. First, the geographical distance, dist(rec_1, rec_2) and the time difference, t_{rec_1,rec_2}, between two sequential records, rec_1 and rec_2 are calculated, such that the speed, v_{rec_1,rec_2}, between the two locations can be obtained, as shown in equation 3.16:

$$v_{rec_1,rec_2} = \frac{\text{dist}(rec_1, rec_2)}{t_{rec_1,rec_2}} \tag{3.16}$$

The speed is compared with a threshold speed of v_{th}=430 km/h, which is known as the maximum speed of urban rail transport. The record will be then removed if the speed exceeds the threshold speed, i.e. $v_{rec_1,rec_2} > v_{th}$. For example, in Figure 3.8, the average speed between p_3 and p_4 can be calculated as $v_{3,4}$, and if $v_{3,4} > v_{th}$, p_4 will be regarded as an outlier and will be removed.

After that, the user trajectory is resampled at a one-minute interval. As shown on the left of Figure 3.9, although the moving path is clear, it is difficult to know the locations where the user stays long (as shown on the right of Figure 3.9). A new user moving trace with a fixed sampling frequency is obtained by applying the resampling technique, as shown in Figure 3.10. From this figure, there are three areas where the GPS points are densely distributed, located at the top left, middle and bottom right in this figure, respectively. Since the points are collected at a fixed time interval, this means that the user spends more time in these areas. After resampling, the trajectory of a user can be denoted by a set of locations expressed as $trace = p_1, \cdots, p_n$. The resampling ensures the subsequent analysis and greatly reduces the data volume. The coordinates of latitude and longitude during each time interval i are averaged to obtain the average position $\left(\overline{\text{lat}_i}, \overline{\text{lon}_i}\right)$ during this interval, as shown in equation 3.10 and equation 3.11.

3.3.2 POI Extraction

As introduced in section 3.2.2, DBSCAN is employed to extract the POIs. Based on the experiment results, the suitable value of the two parameters in the DBSCAN algorithm, Eps, and MinPts, are determined through experiments. The value of Eps is set to 0.01, and the MinPts is set to 10. Figure 3.11 shows an example of extracting POIs from a pre-

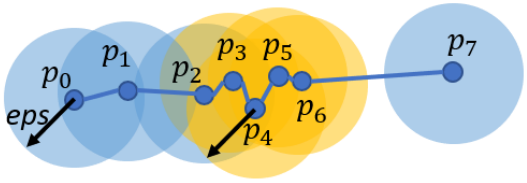


Fig. 3.11 An example of POI extraction

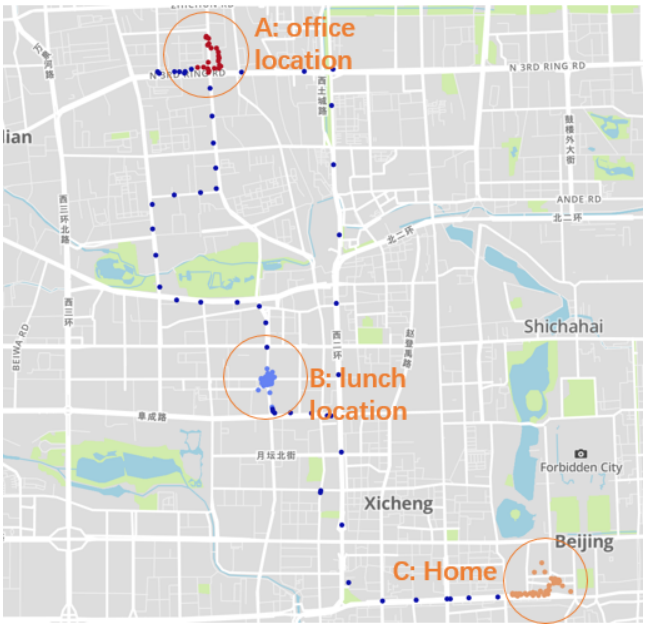


Fig. 3.12 POIs of a user

processed user trajectory. For the location p_1, if assuming MinPts=3 and the Eps as shown in the figure, it can be seen that the total number of points within the range Eps is one, i.e. $N_{p1} = 1 < MinPts$, thus p_1 is considered as noise. In contrast, for the location p_4, by having $N_{p4} = 4 > MinPts$, p_4 is a core point, and finally, the area in yellow is considered as a POI. Figure 3.12 shows the POIs of a user. Three POIs are obtained, shown in red, light blue, and orange, respectively. By combining the time that the user visits these POIs and the geographic features of these areas, it is reasonable to infer that the three POIs, from top to bottom in the figure, are office location, lunch location, and home, respectively. Different types of POIs have been extracted, meaning that the POI extraction algorithm can effectively reflect the user's daily activities.

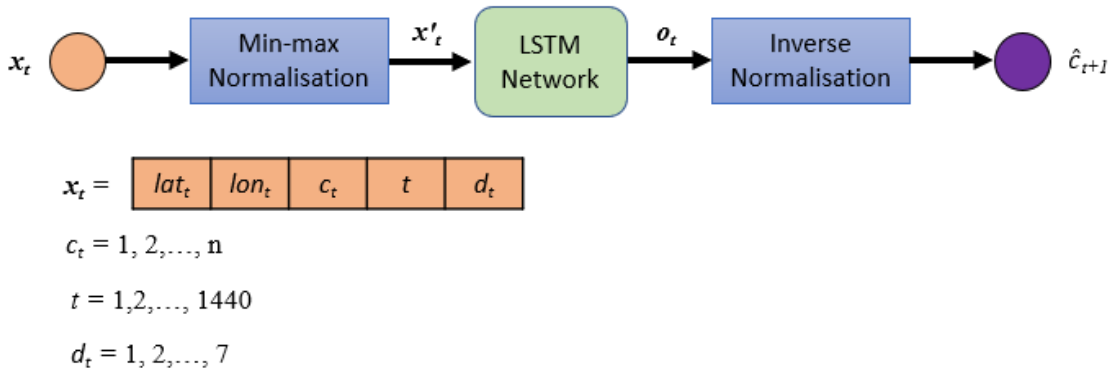


Fig. 3.13 Simplified mobility prediction framework using LSTM

3.3.3 User Mobility Prediction Using LSTM

Figure 3.13 shows the simplified LSTM network framework for mobility prediction. The LSTM network takes a vector as input. For each sample, the length of the input vector is five, representing five features which are extracted in previous sections which can be denoted as in equation 3.12, where lat_t and lon_t are the latitude and longitude of GPS coordinates obtained from section 3.3.1, c_t is the label of the POI extracted in section 3.3.2, t represents the time in minute in a day, from 0:00 to 23:59, and 1,440 minutes in total. d_t denotes the day of week, which is an integer in the range [1,7], representing Monday to Sunday, respectively. All features of $\mathbf{x}_t$ are normalised by min-max normalisation, which makes each feature ranges between 0 and 1, as indicated in Figure 3.13.

The output of the LSTM network is a scalar, denoted as $\widehat{\mathbf{c}}_{t+1}$, which represents the predicted POI label in the next time interval. An LSTM network is trained for each user. In addition, the pre-processed dataset is divided into a training set and test set, which accounts for 80% and 20%, respectively.

To evaluate the performance of the LSTM network, the prediction accuracy defined in equation 3.15 is employed.

To show the superiority of the LSTM, an ANN network is trained which has the same input and output settings as the LSTM network. Ten users are selected, and one ANN and one LSTM network are trained for each user, respectively. The accuracy of each model is

calculated, and the accuracy of the ten ANN networks and ten LSTM networks are averaged to obtain the average prediction accuracy of ANN and LSTM networks, respectively.

To illustrate the prediction process, one user's GPS log is taken from the Geolife dataset. The log includes the user's GPS records within half a year. The log is pre-processed according to the instructions in Section 3.2.1, with a sampling frequency of ten minutes. Figure 3.12 presents the pre-processed trajectory of a day, which shows three POIs during that day. By conducting pre-processing and POI extraction on the log, five POIs are identified in total. Based on existing data, the user spent around 62.5% of the total time staying in one of the five POIs. The LSTM aims to predict whether and which POI the user will be at in the next time interval. An LSTM network is trained following Section 3.2.3, and the user log is divided into a training set and a test set, which accounts for 80% and 20%, respectively. For comparison, an ANN is also trained. The ANN has two hidden layers, and each layer has ten neurons as well as the LSTM network. The prediction accuracy of LSTM network and ANN is recorded respectively.

To evaluate the average prediction performance, ten users are randomly selected. One LSTM network and one ANN is trained for each individual. Therefore, ten LSTM networks and ten ANNs are built in total. The average prediction accuracy of the ten LSTM networks and ten ANNs is calculated. The results show that the average prediction accuracy of ANN achieves 54.9%, whereas the LSTM network achieves 79.7%. The LSTM network improves the prediction accuracy by 45.2% compared with ANN.

Chapter 4

Traffic Feature Analysis

In the last chapter, the POI extraction, along with the input feature extraction used, are two representatives in feature extraction. There are various approaches to feature extraction. In this chapter, the process of feature extraction will be introduced. A feature extraction method is proposed and utilised in a proposed Twitter traffic prediction framework to show its effectiveness. This chapter proposes a low complexity data pre-processing strategy, which utilises statistical analysis to extract the traffic pattern features. Then the extracted features are used to train an LR model. By validating with real-world mobile traffic data, the proposed strategy efficiently improves the LR prediction accuracy.

4.1 Introduction

In recent years, the number of Online Social Network (OSN) users has increased rapidly. OSN applications have become a vital part of people's daily lives. The OSN applications, including Facebook, Twitter, Wechat, WhatsApp, Instagram, and Weibo, have billions of users in summary. The average time per day spent on social media is also increasing. Users in the U.S. spend approximately one hour in OSNs every day, where this amount reaches 4 hours in the Philippines [150]. This trend has attracted many research efforts on the study of OSN application-specific mobile network traffic [151–153], which has shown potentials in

solving people-related issues, such as sentiment analysis [154, 155], election result prediction [156] and traffic event detection [157].

Most existing works about the study of OSN application-specific mobile network traffic focus on exploiting the content in the OSN records. For example, geo-tagged tweets were used to detect the quality of experience complaints [151] and core network failure [152]. Guo and Zhang [153] applied natural language processing techniques based on Twitter data to uncover blackspots in 4G networks in London. However, the prediction of OSN application-specific traffic is still new.

The authors in [12] analysed the network traffic status and subscriber behaviour characteristics. They proposed the feasibility to design the pricing strategy, protocol, and conduct resource and spectrum management. In [80], a regression-based method was proposed, which combines the network key performance indicators to predict network traffic. In [158] and [124], ANN was employed. Similarly, Hua et al. [123], Azzouni and Pujolle [159] proposed a deep learning approach. They used the LSTM network to predict the traffic in telecommunication networks. Nevertheless, all these research works have focused on the aggregate traffic loads generated by all applications while ignoring the traffic characteristics of OSN applications. These existing prediction techniques may not work well, considering the high dynamics in OSN traffic.

Among OSN applications, Twitter has more than 300 million monthly active users [160]. Moreover, Yang et al. [161] have proved that the variation of Twitter traffic can be used to precisely analyse the mobile network traffic trend and the mobility of the population.

In this work, to fill the gaps above, the temporal characteristics of Twitter traffic is studied, and a Twitter traffic prediction framework is proposed which combines statistical analytics and machine learning techniques. In the framework, the statistical analysis act as a part of the pre-processing stage to extract daily Twitter traffic pattern features and filter out the outliers. Then the LR approach is applied to fit the Twitter pattern. Prediction of Twitter traffic in the central London area are discussed to validate the proposed method. Compared with deep learning methods, my approach has low complexity and does not rely on a big dataset. In

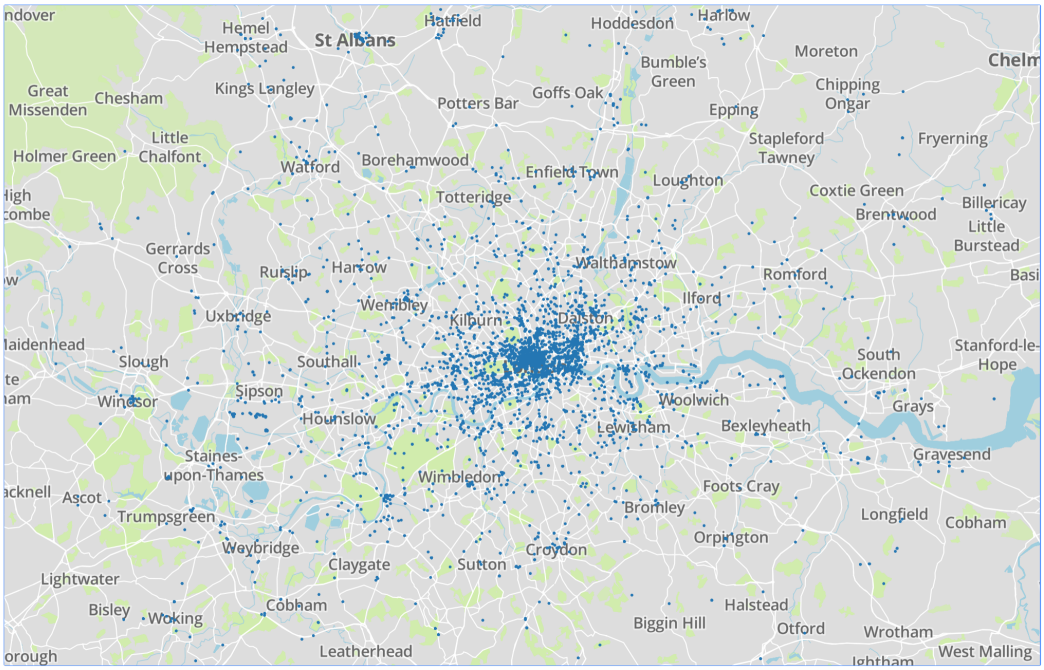


Fig. 4.1 All tweets posted on 15th February 2016

addition, numerical results show better prediction performance than the conventional LR method and neural network.

4.2 The Twitter Dataset and Preliminary Analysis

4.2.1 The Twitter Datasets

In order to conduct the Twitter traffic prediction of Greater London and surrounding suburbs areas, all tweets between 15th and 28th February 2016 in this area were collected and pre-processed for the analysis. Since no important event happened during these two weeks (14 days), the data is representative in representing users' daily behaviours and activities. Each tweet was time-stamped and geo-tagged to investigate the distribution of the data. For example, Figure 4.1 shows the tweets posted on 15th February 2016 in Greater London and surrounding suburbs areas. A total of 8,192 samples were collected, and it can be seen that most tweets are concentrated in central London. In general, the number of tweets varies at different times of the day. The number of tweets in the two weeks from 15th to 28th February

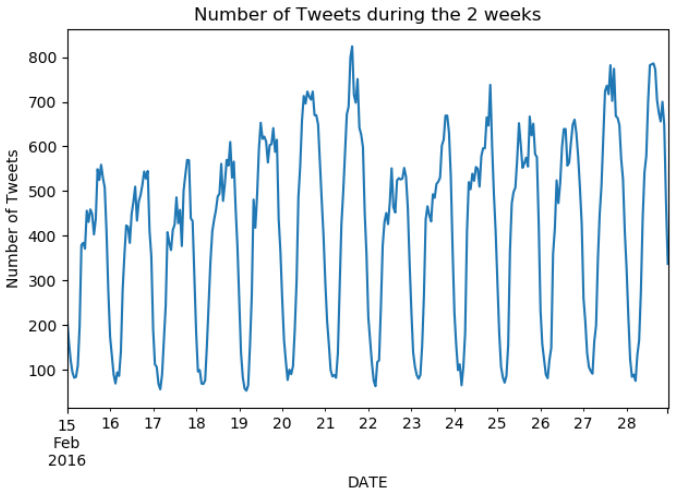


Fig. 4.2 Number of Tweets in two weeks

2016 is displayed in Figure 4.2. It can be seen from Figure 4.2 that the number of tweets periodically changes, and basically, fewer tweets are posted at midnight while more tweets in the daytime. The similarity of the posted tweets numbers among each day is investigated by using the correlation analysis as follows, such to explore the data properties for the traffic prediction.

4.2.2 The Preliminary Analysis

The correlation analysis is conducted in this section showing the tweets data of each day is similar such that the numerical and statistical analysis can be conducted based on the two weeks' data. The similarity of each day's number of tweets can be observed in Figure 4.2, and the details of each day's data are shown in Figure 4.3 by comparing the data changing on 15th and 16th February 2016. It can be seen that peak time appears from 9 a.m to 9 p.m, indicating that users are active from the afternoon to the evening. Valley time occurs at other times, which conforms to the users' sleeping habits.

To validate the regularity of Twitter traffic, Twitter traffic is analysed in the frequency domain. Fast Fourier Transform (FFT) is applied to convert the Twitter traffic from the time domain into the frequency domain. The aim of applying FFT is to observe the Twitter traffic in the frequency domain, which is helpful to extract the periodic features that are not obvious

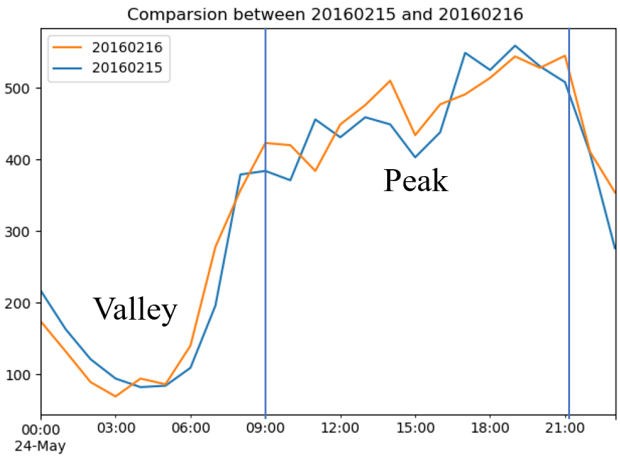


Fig. 4.3 Data comparison between the 15th and 16th February 2016

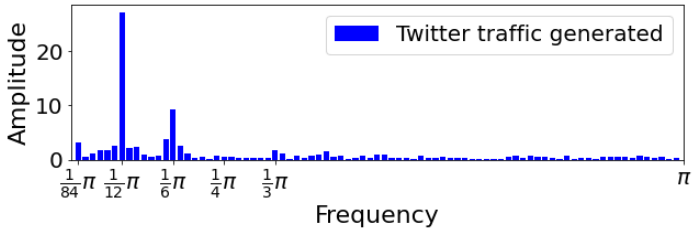


Fig. 4.4 DFS of Twitter traffic during a week

under the time domain observation. For example, the daily regularity is easy to be observed, as shown in Figure 16. However, any other underlying regularities are difficult to be observed directly in the time domain. The FFT is defined as

$$\Gamma_l(k) = \mathrm{FFT}[l[t]] = \sum_{t=0}^{T-1} l[t] \mathrm{W}_T^{kt} \tag{4.1}$$

where T is the number of hours in one week (7 days × 24 hours =168), $\mathrm{W}_T = \mathrm{e}^{-j\frac{2\pi}{\mathrm{T}}}$, j is the imaginary unit. The Discrete Fourier Series (DFS) of the Twitter traffic in a week is obtained by applying FFT, as shown in Figure 4.4. In this figure, the highest peak occurs at the frequency of $\frac{1}{12}\pi$, which corresponds to one day in the time domain. This means that the Twitter traffic shows the strongest regularity with the period of one day, which is consistent with the previous 'peak-valley' observation during a day.

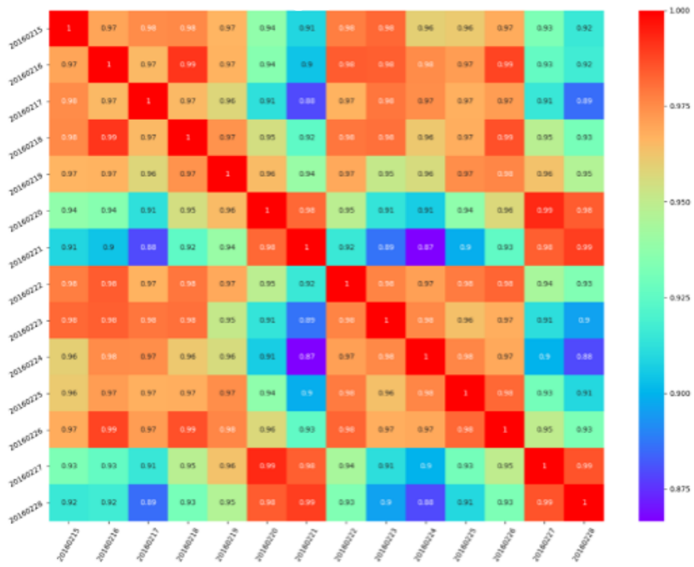


Fig. 4.5 The Pearson correlation heatmap for 1-hour interval

To further find out the relationships among these data, the Pearson correlation coefficient is employed to assess the correlations between each day's tweets data. The Pearson correlation is defined as:

$$\rho_{l_1,l_2} = \frac{\operatorname{cov}(l_1,l_2)}{\sigma_{l_1}\sigma_{l_2}} \tag{4.2}$$

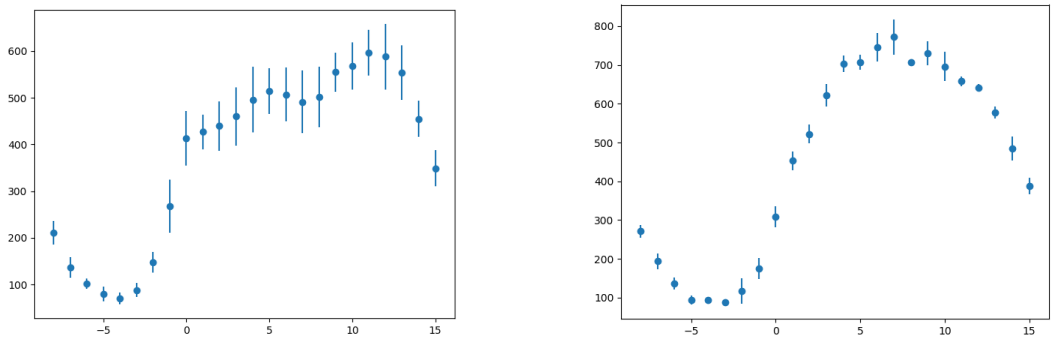
where σ_{l_1} is the standard deviation of l_1 and σ_{l_2} is the standard deviation of l_2, $\operatorname{cov}(l_1,l_2)$ is the covariance between l_1 and l_2, which is defined as:

$$\operatorname{cov}(l_1,l_2) = \mathrm{E}[(l_1-\mu_{l_2})(l_1-\mu_{l_2})] \tag{4.3}$$

with μ_{l_1} being the mean of l_1, and $\mathrm{E}[\cdot]$ represents the expectation.

The Pearson correlation range in equation 4.2 is from -1 to 1, where -1 indicates entirely negative correlated, 0 means no correlation, and 1 represents entirely positive correlated. By conducting Pearson correlation analysis, the similarity between Twitter traffic patterns, such as similarity between daily Twitter traffic patterns, can be obtained. The correlation helps to infer the possible future Twitter traffic patterns.

The Pearson correlation between any two days' pattern is calculated and described as a heatmap, as shown in Figure 4.5. Figure 4.5 shows that most correlation values are



(a) Pattern during weekdays (b) Pattern during weekend

Fig. 4.6 Variation of patterns for (a) weekdays and (b) weekend

larger than 0.9, and all of them are higher than 0.87, which means the curves are highly similar to each other. This characteristic can be concluded as **Remark 4-1**. Moreover, the correlations between weekends, i.e., two Saturdays (20th and 27th February), are higher than the correlations between a weekday and a weekend, indicating the data patterns of weekends have higher similarity to each other than weekdays. Figure 4.6 shows the pattern variability of weekdays and weekends, respectively, where the points represent the mean value. The vertical lines represent the standard deviation of the number of tweets. From this figure, it is noticed that during peak time, the traffic on weekdays has a higher standard deviation than weekends, indicating weekdays' traffic pattern has higher fluctuation than weekends' during peak time. This can be summarised as **Remark 4-2**.

Remark 4-1: The Twitter Traffic pattern shows periodicity and high similarity on a daily basis.

Remark 4-2: The Twitter traffic pattern has burstiness, it has a high deviation from day to day, even during the same time period, and the deviation is higher on weekdays than weekends.

According to the analysis above, the daily pattern can be divided into two parts: the upper half of the curve (Peak times) indicates users are more active, whereas the lower half (Valley times) indicates users are inactive. They are denoting the upper half as the peak period and the lower half as the off-peak period. During the off-peak period, the number of tweets sent

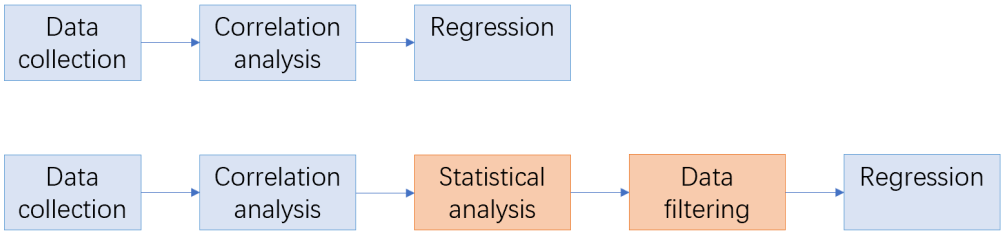


Fig. 4.7 Flow chart of the traffic prediction processes

is stable, and the variation is small, while during the peak period, the number of tweets can fluctuate within a certain range.

In the following studies, the Twitter traffic prediction is discussed by using a combination of statistical and numerical analysis based on the correlation properties of the everyday Twitter traffic.

4.3 The Twitter Traffic Prediction Framework

In order to estimate each hour's Twitter traffic on both weekdays and weekends, according to the correlation analysis above and the results in Figure 4.6, the periodicity can be found. Thus it is straightforward to model the twitter traffics by fitting all collected data with a regression method, as demonstrated at the top of Figure 4.7. For example, a polynomial function can be achieved to represent the Twitter traffic on both weekdays and weekends. However, due to the uncertainty, i.e., high deviation of the Twitter traffic pattern, direct use of the regression method on all collected data may not be accurate for Twitter traffic estimation.

This issue is resolved in this study by conducting the statistical analysis of the collected data before the regression process, such that significant information in the means of statistical remains to improve the accuracy of the regression results. The basic estimation process is illustrated at the bottom of Figure 4.7.

The analysis of the new Twitter traffic estimation approach is discussed as follows.

4.3.1 Statistical Analysis of Twitter Traffic

The purpose of the statistical analysis is to eliminate the effects of the uncertainty or outliers of the collected Twitter traffic data. It can be seen from Figure 4.6 that the data traffic range changes during peak and off-peak times on both weekdays and weekends. For example, on weekdays, the traffic is roughly from 50 to 300 during off-peak times, which is quite small compared to those during peak times from 400 to 600. Consequently, it is much more important to discuss the peak time data traffic than the off-peak time due to the practical requirement. It can be seen from Figure 4.6 that the collected traffic data varies in each hour during the peak time of both weekdays and weekends, which can be quantified by the variation ratio defined as

$$\eta = \frac{l_{\max}[t] - l_{\min}[t]}{\bar{l}_t} \tag{4.4}$$

where $l_{\max}[t]$ and $l_{\min}[t]$ are the upper and lower limit of the statistical data in the t hour, respectively; $\bar{l}_t$ represents the average value of the number of tweets in t.

As a result, the largest variation ratio during peak time on the weekdays is about η=30%, indicating that uncertainty or outliers should be considered before conducting the regression analysis for traffic prediction.

The uncertainty or outliers of the traffic data can be processed by using the statistical analysis method, including two steps as follows:

Step 1: Divide the number of tweets posted during the same periods of different days into intervals, and then calculates the occurrence probability of each interval by using

$$P_i(t) = \frac{n_i(t)}{N_i(t)} \tag{4.5}$$

where $i = 1, 2, \cdots$ represents the different intervals, $n_i(t)$ and $N_i(t)$ represent the occurrence number and the total number of counts during the t hour, respectively.

Step 2: Only the data within the interval with the highest occurrence probability will be used for further regression analysis.

For example, a total of 14 days' data are collected, and the first 11 days' data are considered for the training purpose of the Twitter traffic model. The peak time traffic data

Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
100	0	0.125	0.5	0.875	1	0.75	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
200	0.375	0.875	0.5	0.125	0	0.25	1	0.125	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
300	0.625	0	0	0	0	0	0	0.75	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.125
400	0	0	0	0	0	0	0	0	0.5	0.25	0.25	0.125	0	0	0	0	0.125	0	0	0	0	0	0	0.75
500	0	0	0	0	0	0	0	0.125	0.375	0.625	0.625	0.625	0.75	0.625	0.375	0.625	0.25	0.125	0	0	0	0.125	0.875	0.125
600	0	0	0	0	0	0	0	0	0.125	0.125	0.125	0.25	0.125	0.25	0.5	0.25	0.625	0.625	0.625	0.5	0.75	0.5	0.125	0
700	0	0	0	0	0	0	0	0	0	0	0	0	0.125	0.125	0.125	0.125	0	0.25	0.375	0.5	0.125	0.375	0	0
800	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.125	0	0	0
900	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(a) Weekdays' data

Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
100	0	0	0	0.667	0.667	1	0.333	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
200	0	0.333	1	0.333	0.333	0	0.667	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
300	1	0.667	0	0	0	0	0	0	0.667	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
400	0	0	0	0	0	0	0	0	0.333	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.333
500	0	0	0	0	0	0	0	0	0	1	0.333	0	0	0	0	0	0	0	0	0	0	0	0.667	0.667
600	0	0	0	0	0	0	0	0	0	0	0.667	0.333	0	0	0	0	0	0	0	0	0	1	0.333	0
700	0	0	0	0	0	0	0	0	0	0	0	0.667	0.333	0.667	0	0	0	0.333	0.667	1	1	0	0	0
800	0	0	0	0	0	0	0	0	0	0	0	0	0.667	0.333	1	0.667	1	0.667	0.333	0	0	0	0	0
900	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.333	0	0	0	0	0	0	0	0
1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(b) Weekend's data

Fig. 4.8 The occurrence probability of tweets described in intervals of (a) weekdays and (b) weekends

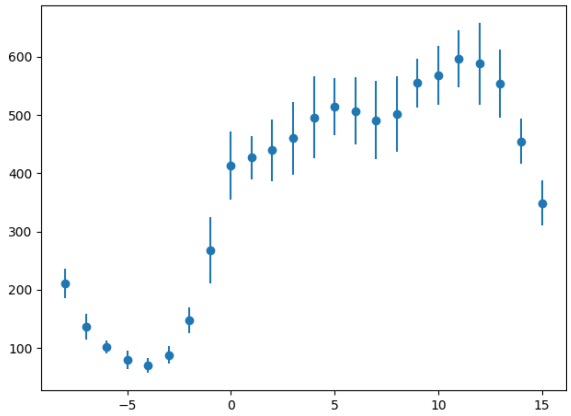


Fig. 4.9 The pattern with statistical analysis on weekdays

are divided into ten levels, and each level has an interval of traffic set as 100. Therefore, the number of tweets located at different intervals in the 11 days can be calculated by using equation 4.5. The results are shown in Figure 4.8, where the index '100' represents the maximum value of the interval, i.e., 100 represents 0-100, 200 represents 100-200; the boxes in blue represent the highest occurrence probability of tweets in an hour. The pattern variability of the weekdays' data from the statistical analysis is shown in Figure 4.9, where the points represent the mean value and the vertical lines represent the standard deviation of the number of tweets. Compared with the results of Figure 4.6(a), it is more evident that the number of tweets during 14:00 and 15:00 decreases and users are the most active during 19:00 and 20:00, due to the working hours and break time, respectively.

In the next section, the LR will be applied based on the processed data by using the statistical analysis above for the prediction of the tweets traffic data.

4.3.2 Twitter Traffic Prediction Using Machine Learning Techniques

In order to represent the variation of Twitter traffic, a polynomial function is applied as

$$l[t] = \alpha_0 + \alpha_1 t + \cdots + \alpha_n t^n \tag{4.6}$$

where $\alpha_i fori = 0, 1, \cdots$ are the coefficients of the polynomial function, $l[t]$ represents the Twitter traffic at the t hour of the day. It is known that when determining a maximum order n,

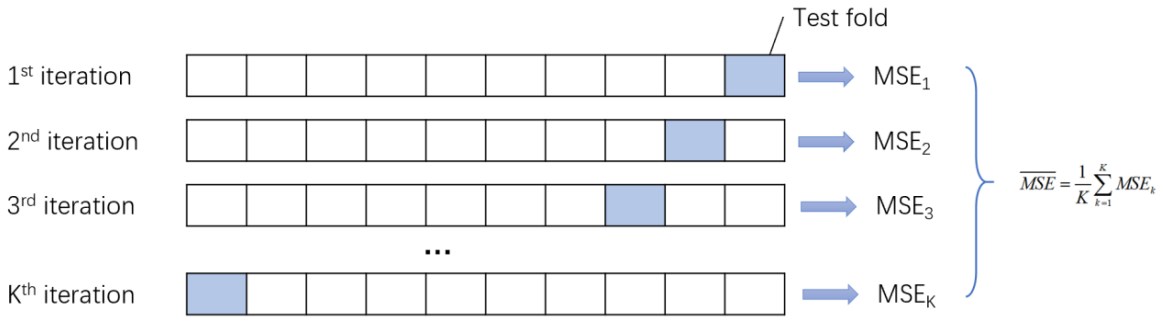


Fig. 4.10 The process of the K-fold cross-validation

the polynomial function can be determined by using the least square method as:

$$\mathbf{A} = \left(\mathbf{T}^T\mathbf{T}\right)^{-1}\mathbf{T}^T\mathbf{D} \tag{4.7}$$

where

$$\mathbf{A} = [a_0, \cdots, a_n]^T \tag{4.8}$$

$$\mathbf{T} = \begin{bmatrix} 1 & \cdots & t_1^n \\ \vdots & \ddots & \vdots \\ 1 & \cdots & t_N^n \end{bmatrix} \tag{4.9}$$

and

$$\mathbf{D} = \begin{bmatrix} l[t_1] \\ \vdots \\ l[t_N] \end{bmatrix} \tag{4.10}$$

where $t_i, i = 1, \cdots, N$ represent the different times of the traffic data.

In order to determine the maximum order n of the polynomial function 4.6 to conduct the least square evaluation algorithm, the K-fold cross-validation is applied in this stage. The basic process of the K-fold cross-validation can be summarised in Figure 4.10.

In the K-fold cross-validation, the Mean Squared Error (MSE) is applied to quantify the regression error of each fold as

$$MSE_k = \frac{1}{N}\sum_{i=1}^{N}\left[\hat{l}\left[t_i\right] - l\left[t_i\right]\right]^2 \tag{4.11}$$

where $\hat{l}\left[t_i\right]$ is the predicted value of the remaining fold except for the K-1 training folds.

The process circulates K times for each order $n = 1, 2, \cdots, K$ of the polynomial function 4.6, and the value of n is chosen with the minimum mean MSE value computed by

$$\overline{MSE} = \frac{1}{K}\sum_{i=1}^{K} MSE_k \tag{4.12}$$

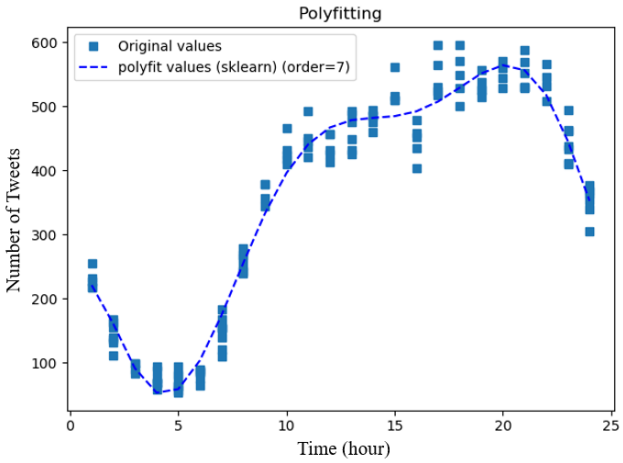
For example, the K value of the Twitter traffic data is chosen as $K = 10$, and the order of the polynomial function representation is obtained as $n = 7$. Consequently, a 7-order polynomial function is applied to represent the Twitter traffic obtained from the statistical analysis of both weekdays and weekends, as shown in Figure 4.11.

Figure 4.11 shows that by applying statistical analysis and filtering, the outliers have been removed. The remaining data points are closely distributed. The curve obtained from LR has managed to fit the Twitter traffic pattern. In addition, the small valley and peak during 14:00 to 15:00 and 19:00 to 20:00 can be noted from the curve, indicating that the details in the pattern have been retained, and fluctuations have been alleviated.

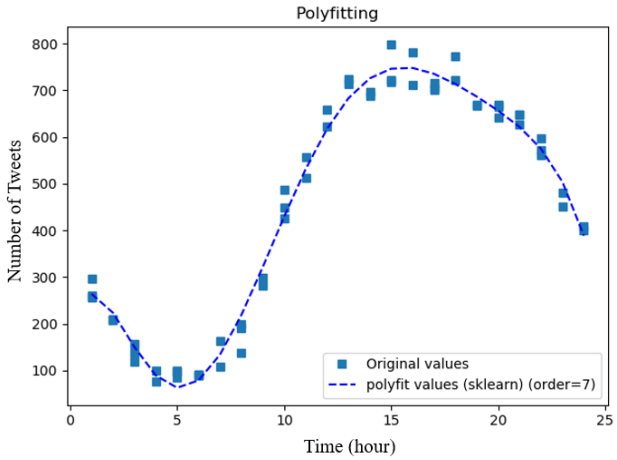
4.3.3 Performance Evaluation

To validate the advantage of the proposed estimation method, the prediction of the Twitter traffic of the remaining three days is conducted for both weekdays (2 days) and weekends (1 day). The direct use of LR on the original data is conducted where, by using the K-fold cross-validation, a 7-order polynomial is used, and the results are shown in Figure 4.12.

The predicted pattern is firstly compared using the proposed method with that without statistical analysis and using a traditional neural network. Figure 4.12 shows the results, where the blue curve is obtained by the proposed method, which combines statistical analysis

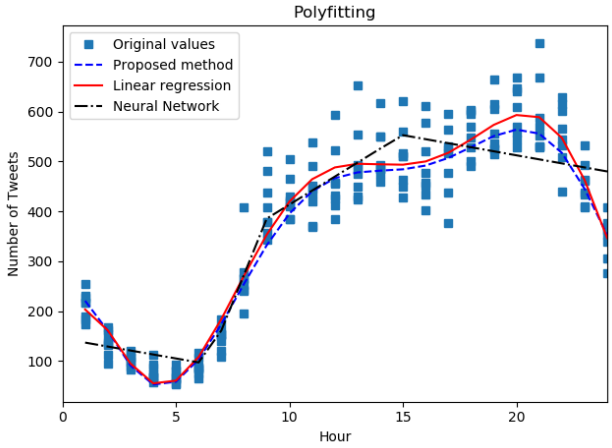


(a) Weekdays' data

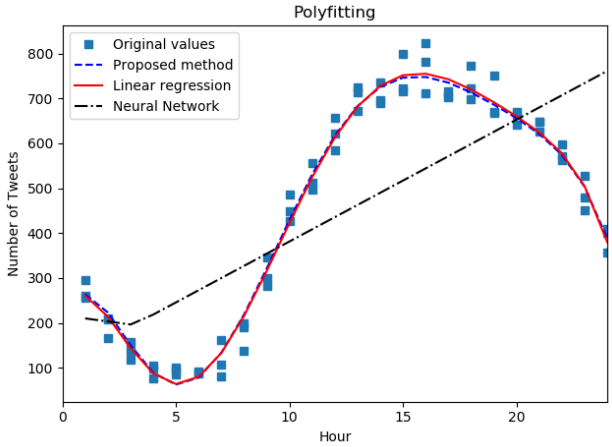


(b) Weekend's data

Fig. 4.11 The seven-order polynomial regression of the Twitter traffic obtained from the statistical analysis for (a) weekdays and (b) weekends



(a) Weekdays' data



(b) Weekend's data

Fig. 4.12 Predicted pattern using the new method and LR for (a) weekdays and (b) weekend

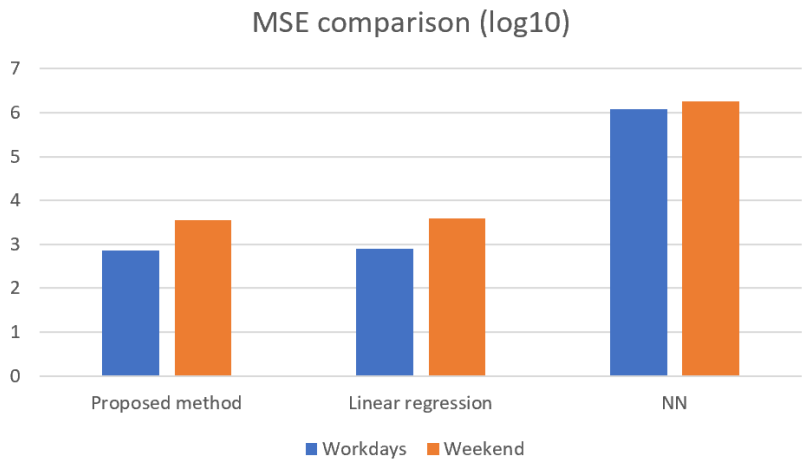


Fig. 4.13 MSE comparison using the proposed method, LR and neural network

and LR. The red curve represents the predicted number of tweets using LR only, without statistical analysis. The two curves are similar, and both outperform the black curve which is obtained by using a neural network. During weekdays, the red and blue curves are pretty close during the off-peak period, and the curve using the proposed method is slightly lower during peak time. For weekend prediction, the patterns predicted by these two methods are quite close. The curve obtained by the neural network performs relatively poor, due to a lack of samples. A neural network requires a considerable size of the dataset to train the model, which is usually much larger than non-machine learning-based methods. In this case, only three days' data, i.e., 72 samples are available, which are far from enough. The lack of samples makes the neural network perform poorly.

To further evaluate the performance of the proposed method, the MSE is adopted as described in equation 4.12, where N represents the number of test samples. Lower MSE means the prediction is closer to the ground truth. Figure 4.13 shows the logarithmic MSE comparison results using the proposed algorithm, LR only, and neural network. In this figure, the MSE obtained from the proposed method is lower under both cases, under both weekdays and weekends. This indicates that the proposed method performs approximately 10% better than that without statistical analysis numerically and much better than using a neural network.

4.4 Conclusion

This chapter studies the temporal characteristics of Twitter traffic and proposes a Twitter traffic prediction framework that combines statistical analytics and machine learning techniques. In the proposed framework, the statistical analysis aims to extract statistical features and pre-process the Twitter data. Then the LR is used to model the temporal Twitter traffic. The proposed framework has several main advantages. The first one is the low computing complexity. Secondly, this method performs well, even if the length of historical data is limited. Lastly, it is easy to keep updated by running statistical analysis and retraining the LR model. Experimental results based on the real-world Twitter traffic dataset collected in central London have validated that the proposed framework has a high prediction accuracy with low computation complexity and low demand for the size of the dataset. In the future, the model versatility, scalability and feasibility will be investigated. For example, the cellular network traffic will be compared to investigate the feasibility of using OSN as a proxy to predict mobile network traffic.

Chapter 5

Mobile Network Traffic Prediction Framework: The ML-TP

In previous chapters, the potentials of deep learning techniques in addressing time series prediction tasks in mobile networks and the importance of feature extraction for time series prediction have been discussed. Based on these achievements, a novel mobile network traffic prediction framework is proposed in this chapter, which integrates both deep learning techniques and feature extraction. This chapter proposes a meta-learning-based Traffic Prediction framework (ML-TP), which can accumulate meta-knowledge from previous cellular level mobile traffic prediction tasks, and adaptively learn to learn the proper prediction model for a new prediction task. For a new traffic prediction task, the ML-TP can quickly generate the proper initial parameters of an LSTM network based on the task's traffic features. Numerical results show that the ML-TP outperforms existing prediction algorithms for the cellular level mobile traffic prediction tasks. Furthermore, the meta-learner in ML-TP significantly improves the base-learner's learning efficiency by leading to about 70% and 80% reduction in the epochs and base-samples required to train their parameters, while keeping similar or even better prediction accuracy.

5.1 Introduction

With the rapid development of mobile communication technologies and applications such as the Internet of Things (IoT), Cloud Computing, and Virtual Reality (VR)/Augmented Reality (AR), the traffic in mobile networks has experienced explosive growth in the past decades. To meet the soaring user demands and reduce the operational expenditure (OpEx), traffic prediction at the cell-level plays a vital role in mobile networks since many mobile applications rely on real-time or near real-time traffic analysis of a radio access network (RAN) [66]. For example, if the future traffic load of a base station (BS) can be forecasted accurately, a sleeping strategy for the base station (BS) can be implemented to reduce the energy consumption [37]. Also, if the downlink mobile traffic data required by terminal devices in a certain area can be predicted and then be stored in edge clouds close to mobile users in advance, mobile subscribers' experience will be greatly improved [93]. Moreover, as the increase of cellular network service diversity and traffic load, traffic prediction also becomes critical for energy optimization [61] and network resource allocation [73].

Mobile network traffic prediction has attracted a lot of attention from both academia and industry. The statistical model-based and shallow machine learning-based prediction methods cannot cope with many practical prediction tasks due to the fact that they rely on some prior knowledge of traffic records to extract mobile traffic features. Although deep learning-based prediction methods are capable of mining the temporal-spatial correlations hidden in traffic patterns, individually training deep learning models for multiple cells is not only time consuming but also sometimes unfeasible since there are not always sufficient historical mobile traffic records available, e.g., for newly built cells and networks. Considering the massive number of cells in 5G and beyond mobile networks, an efficient cell-level traffic prediction method is urgently needed.

In order to fill the above gaps, this chapter presents an early attempt to introduce meta-learning into cell-level mobile traffic prediction, where the knowledge obtained from well-trained prediction models for existing cells or previous traffic prediction tasks will be used to learn the proper prediction model for a new cell or a new traffic prediction task, thus removing the dependency on a large amount of historical mobile traffic records for each cell

or each traffice prediction task. The main contributions of this chapter are summarised as follows:

- Using the real-world mobile traffic records collected in Milan, the characteristics of cell-level mobile traffic in both the time domain and the frequency domain are investigated. Through fast Fourier transform (FFT), it can be found that five main frequency components can characterise the cell-level mobile traffic variations over hours, days, and weeks, hence can be used as the meta-features for a cell-level mobile traffic prediction task.

- The traffic prediction task for each individual mobile cell is regarded as a base-task. By defining the meta-task as learning to learn the proper prediction model for a new base-task according to its meta-features, a novel meta-learning-based cell-level mobile traffic prediction framework (ML-TP) is proposed. The ML-TP adopts a deep long short-term memory (LSTM) network with a fixed structure as the base-learner to forecast a cell's future traffic load. This chapter mathematically proves that each base-task's traffic pattern is dominated by the meta-features and that the well-trained base-learner of a base-task can be seen as a continuous and differentiable function w.r.t. its input. Based on these, this chapter theoretically concludes that transferring the base-learner's parameters of a previous base-task to a new base-task's base-learner will cause limited prediction errors if the two base-tasks have similar meta-features. Then a K-nearest neighbours (KNN) algorithm-based meta-learner is proposed to generate the proper initial parameters for the base-learner of a new base-task based on its meta-features.

- This chapter evaluates the performance of the proposed ML-TP framework through real-world cellular-level mobile traffic prediction tasks. The results show that the ML-TP outperforms existing prediction methods in terms of cell-level mobile traffic prediction accuracy for the same size of training sets. Furthermore, the meta-learner in ML-TP can significantly improve the base-learners' learning efficiency by reducing the

epochs and base-samples required to train their parameters while achieving a similar or even better prediction accuracy.

The rest of this chapter is organised as follows. Section 5.1 reviews the existing works on mobile traffic prediction and meta-learning technology. Section 5.2 describes the mobile traffic traces used, followed by the characteristic analyses of cell-level mobile traffic. The proposed ML-TP is presented in Section 5.3. Section 5.4 evaluates the performance of ML-TP in comparison with representative baseline methods. Finally, Section 5.5 concludes this chapter.

5.1.1 Related Works

In statistical model-based mobile traffic prediction, the time series of mobile traffic are fitted to specific mathematical models, and future traffic loads are predicted based on statistics or probabilistic distributions. Li et al.[162] demonstrated that cell-level mobile traffic loads possess a strong self-similarity and utilised the α-stable model to predict the cell-level mobile traffic fluctuations over time. In [79, 85, 102], the linear autoregressive integrated moving average (ARIMA) model was used to capture the short-term correlation in mobile network traffic. As an extension, the seasonal ARIMA (SARIMA) model was adopted in [86, 103] to improve the ARIMA model on long-term traffic correlation capturing . The authors in [87] proposed an entropy theory-based mathematical model to improve the ARIMA model's prediction accuracy. In addition, Holt-Winter's exponential smoothing model [78], ON-OFF model [88], and sinusoid superposition model [89] were also used to capture the temporal and/or spatial characteristics of mobile network traffic. Though these statistical model-based prediction methods have a relatively low computational complexity, it is difficult for them to estimate the realistic mobile traffic accurately because the real-world irregular traffic patterns are much more complicated than the mathematical models.

In the machine learning-based mobile traffic prediction, linear regression (LR) [80], compressive sensing (CS) [81, 114], support vector regression (SVR) [115], principal components analysis [116], Kalman filtering [118], and Gaussian process [119] were used to forecast the

future trends of cell-level mobile traffic. Nevertheless, these shallow learning-based methods can not cope with many practical prediction tasks because they rely on some prior knowledge of mobile traffic to perform feature extraction.

Powerful deep learning tools have recently been leveraged for mobile traffic prediction. Nie et al. [120] exploited the deep belief network (DBN) based model and the Gaussian model to predict the low-pass and high-pass components of cell-level mobile traffic, respectively. Tian et al. [84] trained a recurrent neural network (RNN) to predict the cell-level mobile traffic by utilising the RNN's capability of capturing the temporal correlations in traffic load time series. Assuming traffic information of neighbouring cells, Qiu et al. [122] proposed a long short-term memory (LSTM) network-based prediction model to forecast the future traffic load of a target cell. With a reduced connection complexity, a model based on a random connectivity LSTM network was proposed in [123] to predict a single cell's traffic load. Wang et al. [124] used local stacked autoencoders and global stacked autoencoders to extract spatial correlations among mobile traffic loads generated in different cells and then trained individual LSTM networks to predict the future traffic loads for different cells. Based on historical traffic loads generated in city-wide networks, a convolutional neural network based prediction model [51] and a convolutional LSTM network-based prediction model [52] were proposed to forecast the spatial distribution of mobile traffic in a city. However, in the existing deep learning-based methods, a specific prediction model must be constructed and trained for each individual prediction task as the mobile traffic patterns associated with different tasks are quite different.

5.2 Dataset Description and Preliminary Analysis

This section introduces the dataset of real-world mobile network traffic records used in this work and presents the characteristic analyses of cell-level mobile traffic in both the time domain and frequency domain.

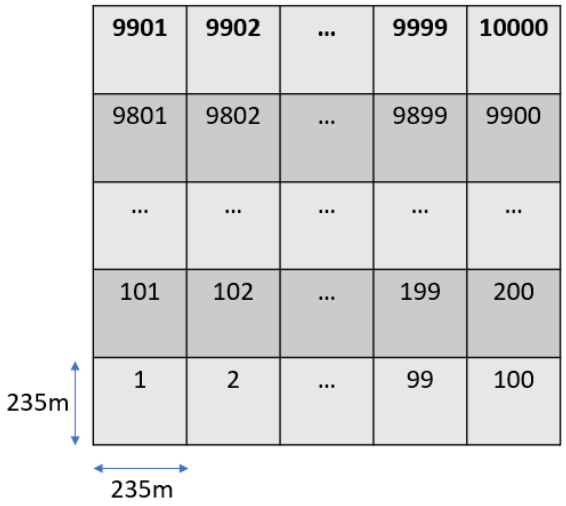


Fig. 5.1 Milan grid

5.2.1 Mobile Network Traffic Trace

This chapterb adopts the mobile network dataset provided by the "Big Data Challenge" program of Telecom Italia [163]. In the dataset, mobile traffic records were collected from 1st November 2013 to 1st January 2014 with a time resolution of ten minutes (about 300 million traffic records in 62 days) over the whole area of Milan. Specifically, the Milan city area is divided into 10,000 grids, each with the same size of 235m x 235m, as shown in Figure 5.1. In each grid, three types of mobile traffic (i.e., short message service, voice call service, and Internet browsing service) were recorded by the operator. Each traffic record contains the time-stamps, the corresponding cell ID, service type, and the volume of data generated. Hereafter, each grid is referred to as a cell, as the grid size approximates to the coverage area of an urban BS. The same grid index as in the original dataset is used.

Denoting the time resolution of the traffic records by Δt, the sum traffic load of the p-th cell during the t-th time interval is given by:

$$l'_p[t] = \sum_{ID_r=p,(t-1)\cdot\Delta t<t_r\leq t\cdot\Delta t} vol_r, r \in \mathbb{R} \tag{5.1}$$

where $\mathbb{R}$ is the set of traffic records, r is an index of traffic record in $\mathbb{R}$, ID_r, t_r, and vol_r are the cell ID, time-stamp, and volume of mobile traffic data of record r, respectively.

The time series of the p-th cell's traffic loads is denoted by a vector, $\mathbf{l'_p} = (l'_p[1], l'_p[2], ..., l'_p[N])$, where N is the total number of total time intervals.The time resolution of traffic records, Δt, is set as one hour following the settings in [52].

In order to analyse mobile traffic characteristics for various cells, the elements of the traffic load vectors are normalised into the range of [0,1] via the min-max normalization method:

$$l_p[t] = \frac{l'_p[t] - min(\mathbf{l'_p})}{max(\mathbf{l'_p}) - min(\mathbf{l'_p})} \tag{5.2}$$

where $max(\mathbf{l'_p})$ and $min(\mathbf{l'_p})$ represent the values of the largest element and the smallest element in $\mathbf{l'_p}$, respectively. Accordingly, the vector $\mathbf{l_p} = (l_p[1], l_p[2], ..., l_p[N])$ is used to record the normalised traffic load vector for cell p.

5.2.2 Characteristics of Cell-Level Mobile Traffic

Figure 5.2 shows the normalised mobile traffic patterns of three different cells, i.e., cells 1684, 1884, and 7121 that are located in the commercial area, the residential area, and the business area, respectively, over the same two weeks. From Figure 5.2, it can be observed that:

Observation 1: Mobile traffic loads of the different cells exhibit various temporal characteristics because they locate in different urban-functional areas. In cell 1684, the mobile traffic was mainly produced from 8:00 to 17:00 in a single day, and the traffic loads on weekends are much higher than those on workdays. In cell 1884, the traffic loads have multiple peaks during 8:00-10:00 and 15:00-17:00, respectively every day, and the difference between workday loads and weekend loads is much less than that in cell 1684. In cell 7121, the traffic loads also have multiple peaks during 9:00-11:00 and 13:00-15:00, respectively every day, but the traffic loads on weekends are much lower than those on workdays.

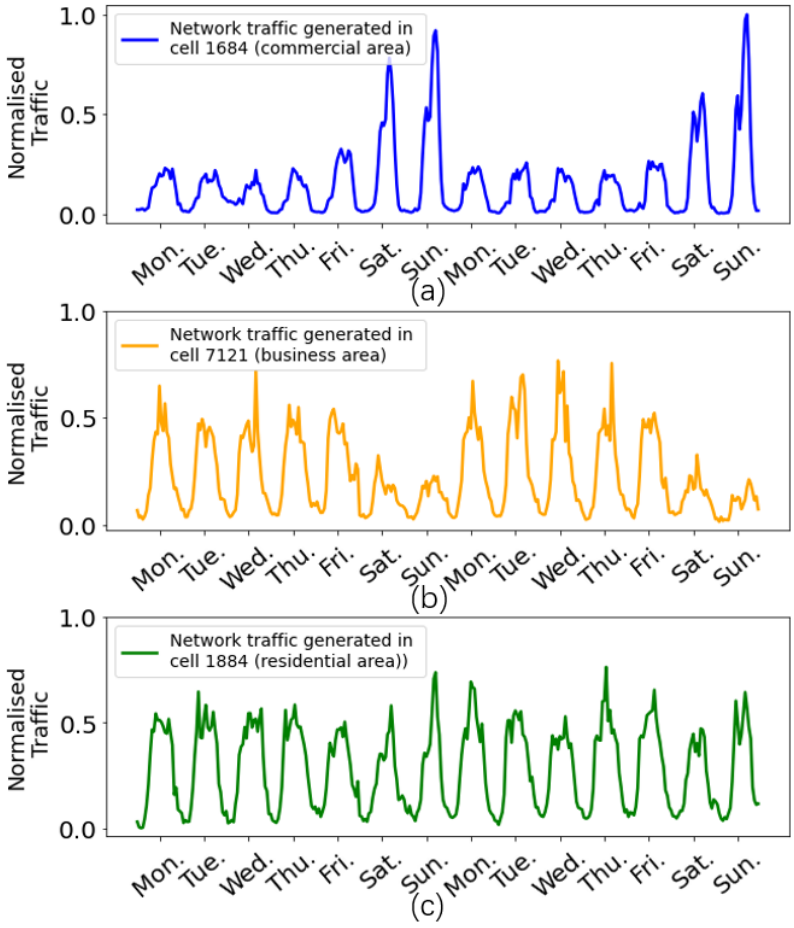


Fig. 5.2 Temporal traffic patterns (normalised, in hours) of cell 1884 (commercial area), 7121 (business area) and 1684 (residential area)

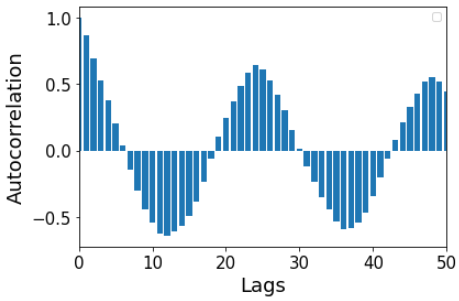


Fig. 5.3 Autocorrelation coefficient of the normalised traffic load vector of cell 1884.

Observation 2: In each cell (including those not shown in Figure 5.2), although the traffic variations are different on different days of the week, they exhibit a weekly periodic pattern.

To quantify the temporal correlation of cell-level mobile traffic, the autocorrelation coefficient of the normalised traffic load vector of cell is calculated as p [164]:

$$cor_{p,k} = \frac{\sum_{t=1}^{168-k} \left(l_p[t] - \bar{\mathbf{l}}_p\right)\left(l_p[t+k] - \bar{\mathbf{l}}_p\right)}{\sum_{t=1}^{168} \left(l_p[t] - \bar{\mathbf{l}}_p\right)^2} \tag{5.3}$$

where $\bar{\mathbf{l}}_p$ represents the mean value of the normalised traffic load vector of cell p, 168 is the length of one week (7 days $\times$ 24).

Figure 5.3 displays the autocorrelation coefficient of the normalised traffic loads in cell 1884. Autocorrelation coefficients of different cells show similar results. From Figure 5.3, it can be observed that:

Observation 3: The normalised cell-level traffic load vector exhibits non-zero autocorrelations in the time domain, and the autocorrelation coefficient reaches peak values when the temporal lag, k, is an integer multiple of 24 hours.

According to **Observation 2**, a discrete periodic signal based on cell p's normalised traffic load vector is constructed firstly, denoted as follows

$$\widetilde{l}_p[t] = \begin{cases} l_p[t], 0 \leq t < 168 \\ l_p[t \bmod 168], t < 0 \; or \; t \geq 168 \end{cases}$$

and then obtain the FFT of this discrete periodic signal:

$$F_p\left(k \cdot \frac{2\pi}{T}\right) = \text{FFT}\left[\widetilde{l}_p[t]\right] = \sum_{t=0}^{T-1} \widetilde{l}_p[t] W_T^{kt}, \; k \in \mathbb{Z} \tag{5.4}$$

where j is the imaginary unit.

Although the above constructed periodic signal may only approximate the actual traffic stream, it allows us to study the features of a cell-level traffic prediction task using historical traffic loads recorded in just one week.

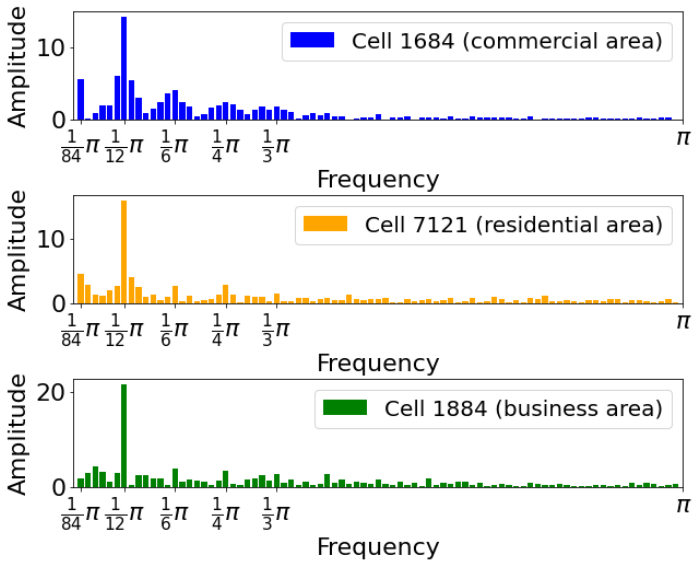


Fig. 5.4 DFS of traffic load for cell 1884, 7121 and 1684

The amplitudes of the FFT results for cells 1684, 1884, and 7121 are shown in Figure 5.4. From Figure 5.4 and the FFT results for other cells are not illustrated in Figure 5.4, the following observation is achieved.

Observation 4: The five frequency components, $\omega = \pi/84$, $\omega = \pi/12$, $\omega = \pi/6$, $\omega = \pi/4$, and $\omega = \pi/3$, which correspond to the periods of one week, one day, 12 hours, 8 hours, and 6 hours, respectively, dominate the frequency-domain characteristics of the normalised traffic load vector of each cell. However, the amplitudes of these frequency components change evidently across different individual cells.

The real and imaginary parts of cell p's five main frequency components form a frequency component vector of size 10:

$$\begin{aligned}\Gamma_p = [\, &\Re\left(F_p\left(\pi/84\right)\right), \Im\left(F_p\left(\pi/84\right)\right), \Re\left(F_p\left(\pi/12\right)\right), \Im\left(F_p\left(\pi/12\right)\right), \Re\left(F_p\left(\pi/6\right)\right), \\ &\Im\left(F_p\left(\pi/6\right)\right), \Re\left(F_p\left(\pi/4\right)\right), \Im\left(F_p\left(\pi/4\right)\right), \Re\left(F_p\left(\pi/3\right)\right), \Im\left(F_p\left(\pi/3\right)\right)\,]\end{aligned} \tag{5.5}$$

where $\Re\left(C\right)$ and $\Im\left(C\right)$ are the real part and the imaginary part of a complex number C, respectively.

In order to examine whether the five main frequency components can characterize the features of traffic load variations for different cells, the Pearson correlation coefficient $\rho_{p,q}$

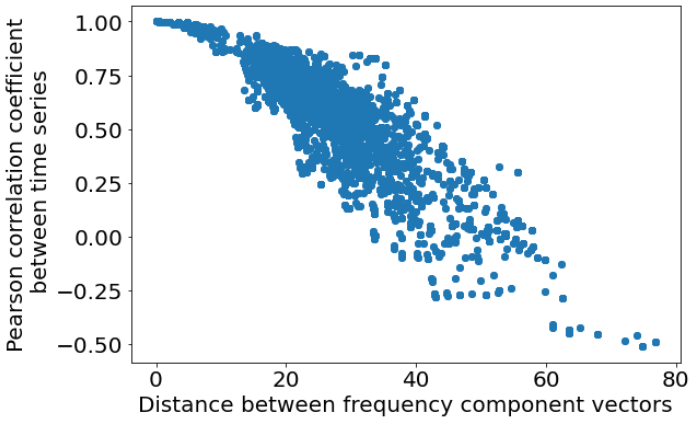


Fig. 5.5 Relationship between the Pearson correlation coefficient of traffic load in the time domain and traffic frequency component vector distance in the frequency domain

for an arbitrary pair of cells, p and q, can be calculated based on their normalised traffic load vectors, $l_p[t]$ and $l_q[t]$ as follows:

$$\rho_{p,q} = \frac{cov\left(\mathbf{l}_p, \mathbf{l}_q\right)}{\sigma_{\mathbf{l}_p} \sigma_{\mathbf{l}_q}} \tag{5.6}$$

where $\sigma_{\mathbf{l}_p}$ is the standard deviation of cell p and $\sigma_{\mathbf{l}_q}$ is the standard deviation of cell q, $cov\left(\mathbf{l}_p, \mathbf{l}_q\right)$ is the covariance between $\mathbf{l}_p$ and $\mathbf{l}_p$, which is defined as:

$$cov\left(\mathbf{l}_p, \mathbf{l}_q\right) = E\left[\left(l_{p[t]} - \mu_{\mathbf{l}_p}\right)\left(l_{q[t]} - \mu_{\mathbf{l}_q}\right)\right] \tag{5.7}$$

where $\mu_{\mathbf{l}_p}$ being the mean of $\mathbf{l}_p$, and $E[.]$ denotes the operational symbol of expectation.

The Euclidean distance between their frequency component vectors, $dist\left(\Gamma_p, \Gamma_q\right)$ is also calculated, which is defined as

$$\text{dist}\left(\Gamma_{l_p}, \Gamma_{l_q}\right) = \sqrt{\sum_{\omega_i \in \omega} \left(\Re\left(F_p\left(\omega_i\right)\right) - \Re\left(F_q\left(\omega_i\right)\right)\right)^2 + \sum_{\omega_i \in \omega} \left(\Im\left(F_p\left(\omega_i\right)\right) - \Im\left(F_q\left(\omega_i\right)\right)\right)^2} \tag{5.8}$$

where ω is the set of main frequency components, i.e., $\omega = \{\pi/84, \pi/12, \pi/6, \pi/4, \pi/3\}$.

For 10,000 randomly selected pairs of cells from the dataset, the Pearson correlation coefficients of their normalised traffic load vectors versus the Euclidean distance between

their frequency component vectors are plotted in Figure 5.5. From Figure 5.5, following observation can be obtained:

Observation 5: The Pearson correlation coefficient of two normalised traffic load vectors is negatively correlated with the Euclidean distance between their frequency component vectors. This indicates that two cells tend to have similar traffic variations in the time domain if their frequency component vectors are close to each other, and vice versa.

Observation 5 implies that the frequency component vector can be used to characterize the features of a cell-level traffic prediction task.

5.3 The Proposed ML-TP

This section first gives an overview of the proposed meta-learning-based cell-level traffic prediction framework, ML-TP. Then, the structures of the base-learners, which are designed for the traffic prediction tasks related to individual cells, and the meta-learner, which is used to improve the learning efficiency of the base-learners, are put forward, respectively.

5.3.1 Meta-learning

Meta-learning is also referred to as “learning to learn" [165, 166]. For a typical supervised learning task ξ, a learning model ς for ξ is usually trained using a set of independent and identically distributed (i.i.d.) samples from the sample space $\mathbb{S}^{\xi}_{train}$ of task ξ. Each sample is depicted by a set of features and is pre-labelled by an unknown task-specific target function F_{ξ}.

The hypothesis space of ς, $\mathscr{H}_{\varsigma}$, is defined as the set of all the possible hypotheses output by ς for arbitrary tasks and arbitrary training sets. Training the learning model ς can be considered as the process of searching for the hypothesis $\hbar_{\varsigma}(\xi)$ that approximates F_{ξ} over the hypothesis space of ς according to the task's sample space, $\mathbb{S}^{\xi}_{train}$.

Due to the learning algorithms adopted (e.g., SVR, decision tree, or deep neural network), the hyper-parameters, or the initial status (e.g., initial connection weights between neurons),

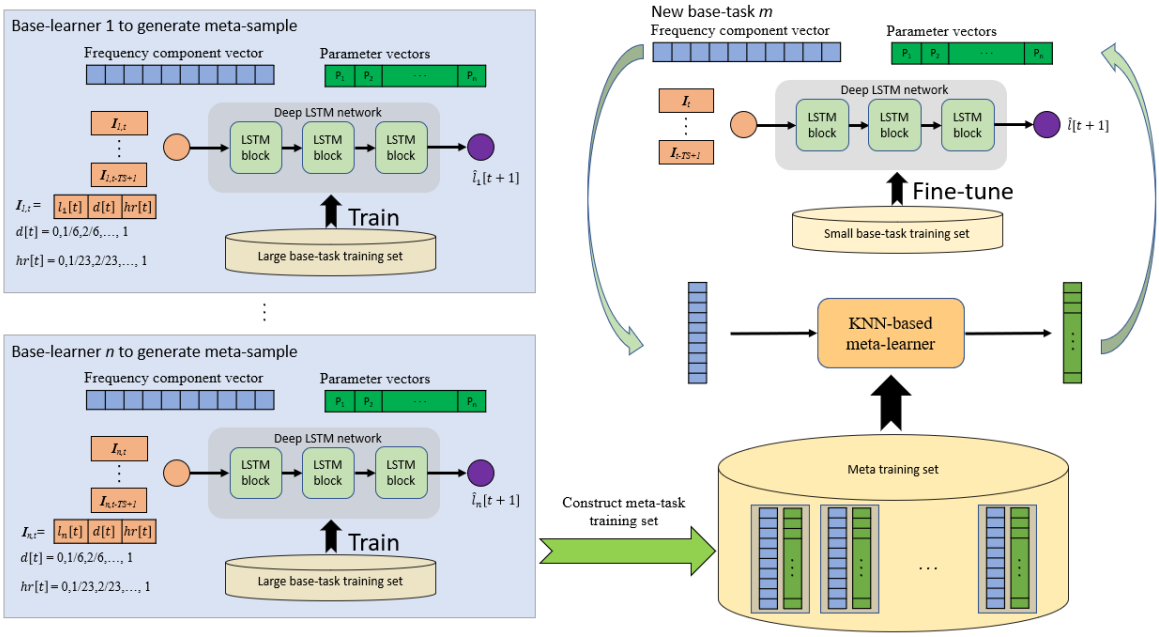


Fig. 5.6 Architecture of the proposed ML-TP framework

the learning model ς often has **biases**. These biases not only affect ς's hypothesis space but also influence the way of ς in searching its hypothesis space to find $\hbar_{\varsigma}(\xi)$.

Different learning tasks, referred to as base-tasks, may prefer different sets of biases embedded in ς. One goal of meta-learning is to find the best set of biases for each base-task according to its meta-features. The learning model handling the meta-learning task (meta-task) is referred to as the meta-learner, while the learning models for base-tasks are called base-learners. In the absence of meta-knowledge, a base-learner has to try all possible sets of biases to find the set of biases that lead to the optimal learning performance. Meta-samples can be obtained by labelling the meta-features of a well-solved base-tasks and used to train the meta-learner. With the accumulation of meta-knowledge from meta-samples, the meta-learner is expected to find the meta-hypothesis, which can produce an approximate best set of biases for the base-learner of a new base-task.

5.3.2 Overview of ML-TP

The diagram of ML-TP is displayed in Figure 5.6. As mentioned before, each cell-level mobile traffic prediction task is regarded as a base-task and present a deep LSTM network

based prediction model with a fixed structure as the base-learner to address it. According to **Observation 5**, a base-task's frequency component vector is defined as its features (meta-features). The initial parameter values, i.e., the initial values of neural connection weights and neural biases, of the deep LSTM network can be further regarded as a base-learner's set of biases.

Specifically, in this figure, the input to the base-learner for task p, i.e., the LSTM network, is $\mathbf{I}_{p,t}$. It is a vector that includes $l_p[t]$, $d[t]$ and $hr[t]$, representing the input features, i.e., traffic load, day of the week and hour in the day. The output of the base-learner is the predicted traffic load at the next hour, i.e., $\hat{l}_p[t+1]$. The meta-learner takes the frequency component vector as input and outputs the initial *PN* parameters for the base-learner.

Obviously, for a certain base-task, the values of neural connection weights and neural biases of the deep LSTM network determine how the base-learner generates the predicted traffic load according to the previous load values, and thus represent the base-learner's hypothesis found in its hypothesis space. All the possible value combinations of neural connection weights and neural biases in the LSTM network structure determines the hypothesis space of a base-learner. In this work, base-learners related to various base-tasks have the same hypothesis space as they have the fixed structure, and a base-learner's set of biases only influences the base-learner's initial searching point in the hypothesis space. It is a reasonable assumption that a base-learner corresponding to a specific base-task will get the highest learning efficiency when its initial searching point approaches this base-task's target function, which is approximated by the hypothesis ultimately found. In other words, the best set of biases of a base-learner will be the final values of neural connection weights and neural biases of the deep LSTM network after it is trained with tremendous base-samples.

This chapter presents a KNN algorithm based meta-learner in ML-TP to handle the meta-task. For any new base-task (cell-level mobile traffic prediction task), the meta-learner will input this base-task's frequency component vector and output the proper initial values of neuron connection weights and neuronal biases for the base-learner according to a set of accumulated meta-samples. This chapter calls this set of meta-samples as the knowledge database of the meta-learner. Each meta-sample in the knowledge database is acquired by

labelling the frequency component vector of a previous base-task with the values of neuron connection weights and neuronal biases in this base-task's well-trained base-learner.

Notations in ML-TP: $\mathbb{S}_{train}^{meta}$ denotes the set of accumulated meta-samples (knowledge database of the meta-learner). $s_{train}^{meta_p}$ in $\mathbb{S}_{train}^{meta}$ denotes the meta-sample related to the p-th mobile cell. Without confusion, $\mathbb{S}_{train}^{meta}$ is used to denote the set of mobile cells generating meta-samples. $\mathbb{S}_{train\_large}^{base_p}$ denotes the large training set with tremendous base-samples for the base-task related to the p-th mobile cell ($p \in \mathbb{S}_{train}^{meta}$). Correspondingly, $\mathbb{S}_{train\_small}^{base_q}$ is the small training set with a few number of base-samples to fine-tune the base-learner of base-task q ($q \notin \mathbb{S}_{train}^{meta}$). $\mathbb{S}_{test}^{base_q}$ is the testing set of base-samples related to base-task q for evaluating the base-learner's performance.

5.3.3 Deep LSTM Network as the Base-learner

Recurrent neural networks (RNNs) allow the information of past inputs to be memorised in the networks' internal states and thereby make them capable of handling input data with historical dependencies. Nevertheless, for a standard RNN architecture, the influence of any input on the network's output will either decay or blow up exponentially when information cycles around the network's recurrent connections. To address this problem, the deep LSTM network, an elegant RNN architecture, has been designed. Deep LSTM network has shown progressive performances in time series forecasting tasks such as language modelling, handwriting recognition, and mobile network traffic prediction [167].

The key component that makes LSTM networks possess the ability to model long-term dependencies is the LSTM memory block. As illustrated in Figure 5.7, each LSTM memory block is a recurrently connected subnet logically, which contains some functional modules called gates. According to their corresponding practical functionalities, these gates are classified as input gate, forget gate, input activation gate and output gate. The input gate controls how much new information flows into the memory block's current state, while the forget gate controls how much information still remains in the memory block's current state through the recurrent connection. Finally, the output gate controls how much information is used to compute the output activation of the memory block and further flows into the rest of

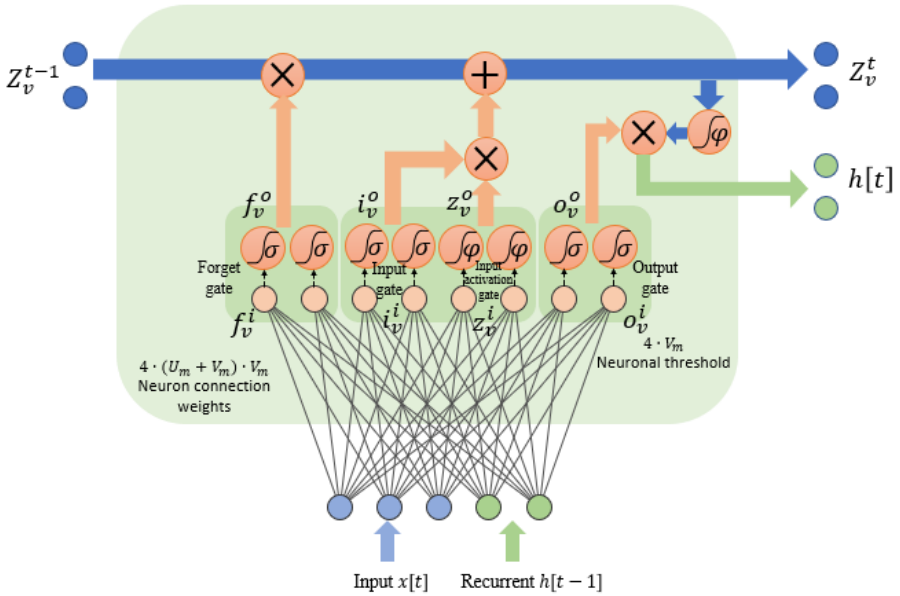


Fig. 5.7 Inner structure of a LSTM network, where the green squares represent the forget gate, input gate, input activation gate and output gate, respectively

the LSTM network. In a basic LSTM memory block, each gate consists of a two-layer neural network where the neuron number of the output layer equals the dimension of the output vector, and the neuron number of the input layer equals the dimension of the input vector plus that of the output vector. In Figure 5.7, $\oplus$ and $\otimes$ denote summation and dot product of two vectors, respectively. $\sigma(x)$ usually takes the sigmoid function, and $\varphi(x)$ usually takes the hyperbolic tangent function as shown in 3.6 and 3.7, respectively.

In this chapter, a multi-layer LSTM network is constructed to act as the base-learner in the ML-TP framework. For specific mobile cell i, this multi-layer LSTM network will continuously input a sequence of input vectors and finally output the predicted mobile traffic load in the next time interval. The length of each input sequence is denoted as the step number, SN. The input sequence is denoted as $[\mathbf{I}_p[t]\ \mathbf{I}_p[t-1], ..., \mathbf{I}_p[t-SN+1]]$. According to **Observation 2** that the traffic load of a mobile cell is strongly influenced by the time, each input vector $\mathbf{I}_p[t]$ is constructed consisting of three attributes: one is the normalised mobile traffic load during the time interval in this cell, $l_p[t]$, and the other two attributes, d_t and hr_t, reflecting the temporal information of time interval t (d_t equals 1/7, 2/7, ..., or 1 denoting Monday, Tuesday, ..., or Sunday, hr_t equals $g/24$ denoting t is during the g-th hour of a day).

M is used to denote the layer number of the proposed LSTM network, use U_m and V_m to denote the dimensions of the m-th layer LSTM memory block's input vector and output vector. Obviously, it can be noticed that $V_m = U_{m+1}$. For the m-th layer LSTM memory block in the LSTM network, there are $4 \times (U_m + V_m) \times V_m + 4 \times V_m$ parameters $(4 \times (U_m + V_m)) \times V_m$ neural connection weights and $4 \times V_m$ neural biases) can be trained in this block as shown in Figure 5.7. Thus, the total number of parameters to be trained, PN, in each base-learner can be calculated as follows:

$$PN = \sum_{m=1}^{M} 4 \times (U_m + V_m) \times U_m + 4 \times V_m \tag{5.9}$$

For an arbitrary mobile cell p, the input sequence is labelled as $[\mathbf{l}_p[t], \mathbf{l}_p[t-1], ..., \mathbf{l}_p[t-SN+1]]$ with this cell's realistic normalised mobile traffic load during time interval $t+1$, $l_p[t+1]$, and transform it into a base-sample. By applying a sliding window with a length of SN to split cell p's normalised traffic load vector, a number of $BSN = N - SN$ base-samples will be built for this cell.

5.3.4 The KNN Algorithm Based Meta-learner

Without meta-knowledge, a base-learner has to be trained with a randomly selected set of biases: random initial values for its neural connection weights and neural biases. For each mobile cell p in $\mathbb{S}_{train}^{meta}$, its base-learner can be trained using the large base-task training set, $\mathbb{S}_{train\_large}^{base_p}$, which comprises all the BSN base-samples related to this mobile cell. By labeling mobile cell p's frequency component vector with the values of neural connection weights and neural biases in its well-trained base-learner, the meta-sample, $s_{train}^{meta_p}$, can be obtained.

As it is challenging to train all the base-learners related to numerous mobile cells with large base-task training sets, whether the proper set of biases for the base-learner of a new base-task can be provided with the help of accumulated meta-knowledge is investigated. Indeed, this chapter can prove the following **Lemmas 1** and **2**, as well as **Propositions 1** and **2**.

Lemma 1: Assume $\mathbf{l}_1 = [l_1[-\infty], ..., l_1[\infty]]$ and $\mathbf{l}_2 = [l_2[-\infty], ..., l_2[\infty]]$ are discrete periodic signals with the same period of T. $\mathbf{l}_1$ and $\mathbf{l}_2$ have some main frequency components at the same set of frequencies, $f_{main} = \{f_1, f_2, ..., f_E\}$, in the range of $[0, 2\pi]$ through FFT, and the sum amplitude of the other frequency components of $l_1[t]$ or $l_2[t]$ in $[0, 2\pi]$ does not exceed η. The vectors $\Gamma_1 = (\Re(F_1(f_1)), \Im(F_1(f_1)), ..., \Re(F_1(f_E)), \Im(F_1(f_E)))$ and $\Gamma_2 = (\Re(F_2(f_1)),$
$\Im(F_2(f_1)), ..., \Re(F_2(f_E)), \Im(F_2(f_E)))$ can be used to record the main frequency components of $l_1[t]$ and $l_2[t]$ in their FFT results, respectively. Then if Euclidean distance between Γ_1 and Γ_2 is σ, then $|l_1[t] - l_2[t]| \leq \sqrt{k} \cdot \sigma + 2 \cdot \eta$, $\forall t \in Z$.

Proof: See Appendix A.

Lemma 2: For an LSTM memory block, whose input vector $\mathbf{x}[t] = (x_1[t], ..., x_U[t])$ and output vector $\mathbf{h}[t] = (h_1[t], ..., h_V[t])$ have the dimensions of U and V, respectively, $(\mathbf{x}[t], \mathbf{x}[t-1], ..., \mathbf{x}[t-SN+1])$ and $(\mathbf{h}[t], \mathbf{h}[t-1], ..., \mathbf{h}[t-SN+1])$ is used to denote its input sequence and output sequence, respectively, for certain number of steps, SN. Then when the LSTM memory block has given values of neural connection weights and neural biases in its input gate, forget gate, input activation gate and output gate, for an arbitrary element $h_{\mathbf{v}}$ in the output vector, $h_v[t]$, $h_v[t-1]$, ..., and $h_v[t-SN+1]$ are continuous and differentiable functions about $x_1[t-SN+1]$, ..., $x_1[t]$, ..., $x_U[t-SN+1]$, ..., $x_U[t]$.

Proof: See Appendix B.

Proposition 1: For a deep LSTM network, which consists of W layers of stacked LSTM blocks, $\mathbf{x}[t] = (x_1[t], ..., x_U[t])$, $(\mathbf{x}[t], \mathbf{x}[t-1], ..., \mathbf{x}[t-SN+1])$, $\mathbf{y}[t] = (y_1[t], ..., y_V[t])$, and $(\mathbf{y}[t], \mathbf{y}[t-1], ..., \mathbf{y}[t-SN+1])$ is used to denote its input vector, input sequence, output vector, and output sequence, respectively. With given neural connection weights and neural biases of this LSTM network, $\mathbf{y}_1[t]$, ..., $\mathbf{y}_V[t]$ are continuous and differentiable functions w.r.t. $x_1[t-SN+1]$, ..., $x_1[t]$, ..., $x_U[t-SN+1]$, ..., $x_U[t]$.

Proof: See Appendix C.

Proposition 2: Suppose $\mathbf{l}_1$ and $\mathbf{l}_2$ are discrete periodic signals with the same period of T. $\mathbf{l}_1$ and $\mathbf{l}_2$ have some main frequency components at the same set of frequencies, $f_{main} = \{f_1, f_2, ..., f_E\}$, in the range of $[0, 2\pi]$ through FFT, and the sum amplitude of the

Algorithm 2 : The KNN algorithm based meta-learner in ML-TP

1: input the frequency component vector Γ_q and the small training set $\mathbb{S}_{train\_small}^{base_q}$ of mobile cell q;
2: input the the set of accumulated meta-samples, $\mathbb{S}_{train}^{meta}$;
3: **for** each meta-sample $s_{train}^{meta_p}$ in $\mathbb{S}_{train}^{meta}$ **do**
4: calculate the the Euclidean distance between Γ_q and the frequency component vector of mobile cell p, Γ_p;
5: **end for**
6: retrieve the K meta-samples in $\mathbb{S}_{train}^{meta}$ whose frequency component vectors have the smallest Euclidean distance with Γ_q;
7: record the set of those K meta-samples as $s_{train}^{meta_{p1}}, s_{train}^{meta_{p2}}, ..., s_{train}^{meta_{pK}}$;
8: **for** each meta-sample $s_{train}^{meta_{pk}}$ in $s_{train}^{meta_{p1}}, s_{train}^{meta_{p2}}, ..., s_{train}^{meta_{pK}}$ **do**
9: transfer the parameter values in $s_{train}^{meta_{pk}}$'s base-learner, i.e., the neural connection weights and neural biases, to the base-learner of mobile cell q;
10: test the predicting accuracy of mobile cell q's base-learner over $\mathbb{S}_{train\_small}^{base_q}$;
11: **end for**
12: retrieve the meta-sample $s_{train}^{meta_{pk^*}}$ in $s_{train}^{meta_{p1}}, s_{train}^{meta_{p2}}, ..., s_{train}^{meta_{pK}}$, whose base-learner's parameters lead to the best predicting accuracy of the base-learner of mobile cell q over $\mathbb{S}_{train\_small}^{base_q}$;
13: output the parameter values in $s_{train}^{meta_{pk^*}}$'s base-learner;

other frequency components of $\mathbf{l}_1[t]$ or $\mathbf{l}_2[t]$ in $[0, 2\pi]$ does not exceed η. The Euclidean distance between $\mathbf{l}_1[t]$'s and $\mathbf{l}_2[t]$'s main frequency component vectors, Γ_{l_1} and Γ_{l_2}, is σ. Then if σ and η are small enough, and a deep LSTM network, which has one-dimensional output vector $y[t]$ and certain number of steps, SN, can accurately predict $l_1[t+1]$, i.e., $y[t] = l_1[t+1]$ using the input vector $\mathbf{x}[t] = (l_1[t], d[t], hr[t])$, then the error of utilising this deep LSTM network to predict $l_2[t+1]$, i.e. $|\hat{y}'[t] - l_2[t+1]|$, using the input vector $x'[t] = (l_2[t], d[t], h[t])$ is bounded by $(\sqrt{k} \cdot \sigma + 2 \cdot \eta) \cdot (1 + \frac{\partial y[t]}{\partial l_1[t]} + ... + \frac{\partial y[t]}{\partial l_1[t-SN+1]})$ for all t in Z.

Proof: See Appendix D.

According to previous observations that the cell-level mobile network traffic varies approximately periodically and is dominated by the five main frequency components, **Proposition 1** implies that, in ML-TP, transferring the values of neural connection weights and neural biases in the base-learner of a previous base-task to a new base-task's base-learner will

cause limited prediction error if the previous base-learner is well-trained and the Euclidean distance between the two base-tasks' frequency component vectors is small. In other words, if choosing these values of neural connection weights and neural biases as the set of biases for the new base-task's base-learner, the new base-learner will have a proper initial status, and its learning efficiency may be improved.

Thus, the Euclidean distance can be defined as the difference between two base-tasks' meta-features and propose a KNN algorithm based meta-learner. For any new base-task q with a limited number of training base-samples, $\mathbb{S}^{base_q}_{train_{small}}$, the meta-learner will first extract its meta-features and calculate the difference between this task's meta-features and the meta-features of each meta-sample in $\mathbb{S}^{meta}_{train}$. Then, the meta-learner will set the label of each meta-sample in the k ones, which have the smallest difference values with the base-task q in $\mathbb{S}^{meta}_{train}$, as the new base-learner's set of biases, respectively, and test the base-learner's performance on $\mathbb{S}^{base_q}_{train_{small}}$. Finally, the meta-learner will choose the label leading the lowest predicting error of the new base-learner on $\mathbb{S}^{base_q}_{train_{small}}$ as the new base-learner's set of biases. The pseudo-code of the KNN algorithm based meta-learner is given in **Algorithm 1**.

5.3.5 Fine-tune the Base-learner for a New Base Learning Task

For any base-task related to a new mobile cell q, as shown in **Algorithm 1**, the proper set of biases for its base-learner will be output by the meta-learner based on its meta-features. Then, the base-learner is fine-tuned using the corresponding base-samples in $\mathbb{S}^{base_q}_{train_{small}}$. With meta-knowledge, the base-learner of a new mobile traffic prediction task is expected to obtain a good prediction accuracy and learning efficiency in terms of fast convergence speed and a small number of training base-samples needed.

5.4 Evaluation on Real-world Mobile Traffic Data

This section conducts comprehensive experiments to investigate the performance of ML-TP for cell-level mobile traffic prediction. Firstly, the experimental settings and performance metrics used are introduced. Then, this section test the influence of two key hyper-parameters, i.e.,

the scale of $\mathbb{S}_{train}^{meta}$ and the selection of k in **Algorithm 1**, on the meta-learner's performance. After that, the prediction accuracy of the ML-TP is compared with several baseline methods. Finally, how the meta-learner can help base-learners improve their learning efficiency in terms of converging speed, and the number of base-examples needed is demonstrated in detail.

5.4.1 Experimental settings and performance metrics

In the dataset, mobile traffic records of some cells during a few time intervals are not successfully collected due to base station failure or data storage error. If the actual traffic load of a mobile cell at a certain time interval is missing, according to the high correlation among adjacent cells, it will be filled using a common method [168] that this missing value is completed by the mean traffic load of the surrounding cells at the same time intervals.

Each base-learner in ML-TP is constructed as a three-layer deep LSTM network, i.e., $M = 3$, and its output vectors of the three layers are 5, 5, and 1 respectively. According to equation 5.9, the total number of parameters to be trained in each base-learner is 428. The step number, SN, of a base-learner is set to be 3. The meta-learner in ML-TP takes the main frequency vector of a new base-task with size 10 as its input, and outputs a vector of size 428 representing the initial parameter values in the new base-task's base-learner.

For each base-task, i.e., the traffic prediction problem of a mobile cell in the dataset, a sliding window with size $SN = 3$ is applied to split its normalised traffic loads and generate the base-samples by labelling each sequence of input vectors with the normalised traffic load in next time interval. Moreover, those base-samples during 24/12/2012-01/01/2013 when the holidays introduced evident mobile traffic abnormalities are abandoned. This secction randomly selects 70% of mobile cells in the dataset to construct the meta-training set $\mathbb{S}_{train}^{meta}$. To generate a meta-sample for each base-task i in $\mathbb{S}_{train}^{meta}$, the base-learner from a randomly selected initial status is trained using all the base-samples generated corresponding to this base-task. This section evaluates the performance of ML-TP on testing base-tasks related to the other 30% of mobile cells. For each testing base-task q, base-samples generated during

11/01/2013-12/15/2013 are used to fine-tune the base-learner and the other ones are used to test the base-learner's prediction accuracy.

This section compares the performance of ML-TP with several methods that are commonly used in time series prediction. These baseline methods include ARIMA [79, 85, 102], Linear Regression (LR) [80], SVR [115] and the conventional LSTM network [122]. ARIMA is an advanced statistical model based method for time series prediction problems, while LR and SVR are representatives of shallow learning methods. Different from these statistical model-based methods or shallow learning methods, the conventional LSTM network is one of deep learning methods. For a fair comparison, the conventional LSTM network structure is the same as each base-learner in ML-TP for each base-task. Different from the base-learner in ML-TP, the initial values of neuron connection weights and neuronal biases of the conventional LSTM network are randomly initialised.

The base-learners in ML-TP and the conventional LSTM networks related to different base-tasks are optimised based on a stochastic gradient-based optimization technique, Adam [148], which is widely adopted in deep learning domain, with the batch size of 8 and the learning rate of 0.001. The mean square error (MSE) loss is chosen as the loss function in the training process.

To evaluate the prediction performance, three metrics, i.e., Root Mean Square Error of Normalised traffic (NRMSE), Mean Absolute Error of Normalised traffic (NMAE), and R-squared (R2) are adopted in the experiments. NRMSE and NMAE reflect how the prediction values approach the real values, while R2 measures the fitting degree between the prediction and ground true values. NRMSE, NMAE, and R2 can be calculated as follows:

$$\mathrm{NRMSE}_p = \sqrt{\frac{1}{N}\sum_{t=1}^{N}\left(\widehat{l}_p[t] - l_p[t]\right)^2} \tag{5.10}$$

$$\mathrm{NRMAE}_p = \frac{1}{N}\sum_{t=1}^{N}\left|\widehat{l}_p[t] - l_p[t]\right| \tag{5.11}$$

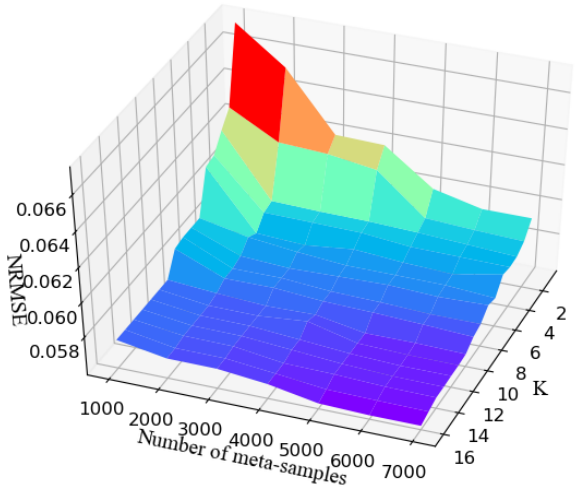


Fig. 5.8 NRMSE over the variation of K of the meta-learner and number of meta-sample

$$\mathrm{R2}_p = 1 - \frac{1}{N} \frac{\sum_{t=1}^{N} \left(\widehat{l}_p[t] - l_p[t] \right)^2}{\sum_{t=1}^{N} \left(\bar{l}_p[t] - l_p[t] \right)^2} \tag{5.12}$$

where N, $\widehat{l}_p[t]$ and $l_p[t]$ represent the total number of samples in calculation, predicted traffic and ground truth traffic, respectively. $\bar{l}_p[t]$ is the average of the ground true values.

5.4.2 Influence of the meta-learner's two key hyper-parameters

For the meta-learner in **Algorithm 1**, there are two key hyper-parameters affecting its performance: the value of K and the scale of $\mathbb{S}_{train}^{meta}$, W, i.e., the number of meta-samples. K determines the number of candidates meta-samples used to choose the optimal set of biases for the base-learner of each new base-task and the scale of $\mathbb{S}_{train}^{meta}$ reflects how much meta-knowledge has been accumulated.

Figure 5.8 presents the average NRMSE achieved by the base-learners of the testing base-tasks over the corresponding testing base-samples when their initial statuses are given by the meta-learner, and they are not fine-tuned by any base-samples, under various values of K and W. Obviously, a smaller average NRMSE value reflects that the meta-learner in ML-TP can provide better sets of biases (initial statuses) for the base-learners of the testing base-tasks. From Figure5.8, it can be noticed that as K rises from 2 to 16 and the number of

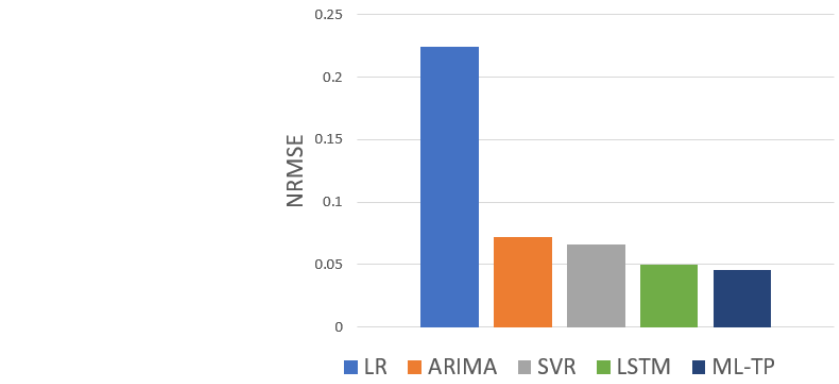
meta-samples rises from 1000 to 7000, the average NRMSE decreases transparently from around 0.067 to lower than 0.057. Moreover, for a certain value of W or K, increasing the value of the other hyper-parameter monotonously elevates the meta-learner's performance in terms of the average NRMSE achieved. These observations can be explained as when more meta-knowledge is accumulated or considering more candidate meta-samples for a new base-task, the meta-learner will have a higher probability to find the previous base-tasks possessing similar traffic patterns with the target base-task and more chances to obtain the proper set of biases for the new base-task's base-learner, which leads to a small prediction error.

An interesting phenomenon in Figure 5.8 is that when K increases from 1 to 16, the average NRMSE first decreases dramatically and then becomes stable under each certain value of W. More specifically, after K exceeds 10, further augment of K seems to introduce little performance improvement for the meta-learner in ML-TP. This is a meaningful conclusion that provides a guide to the hyper-parameter selection in **Algorithm 1**. In the rest of the experiments, the value of K is set to 10 to balance the framework performance and the algorithm complexity.

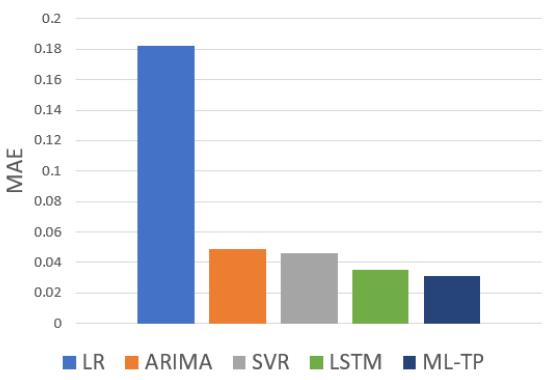
5.4.3 Prediction accuracy of ML-TP and the baseline methods

The prediction performance of ML-TP and the baseline methods are compared over the testing base-tasks. The results of evaluation metrics are shown in Figure 5.9. For each testing base-task q, $\mathbb{S}_{train_{small}}^{base_q}$ is set to contain all the base-samples generating during 11/01/2013-12/15/2013, to fine-tune/train the base-learner in ML-TP and the baseline methods.

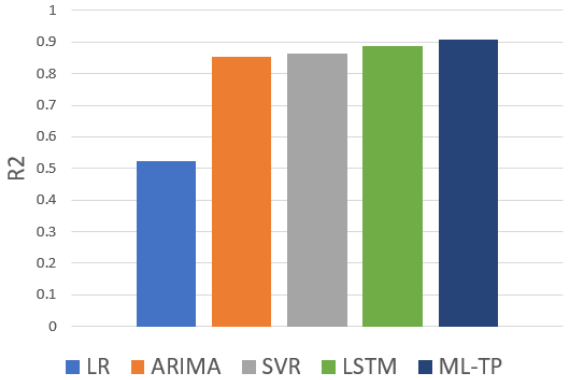
From Figure 5.9, it can be clearly seen that the LR and the ARIMA perform the worst among all the methods. This can be explained as mobile traffic loads have highly nonlinear patterns in the temporal dimension, which makes the future traffic load forecasting a very challenging task and beyond the ability of linear models. The SVR method has the capability of dealing with nonlinearities in mobile traffic loads. Thus it achieves better prediction performance than that of LR and ARIMA. The high complex network architecture enables the conventional LSTM network to have strong ability to represent the nonlinearities in



(a)



(b)

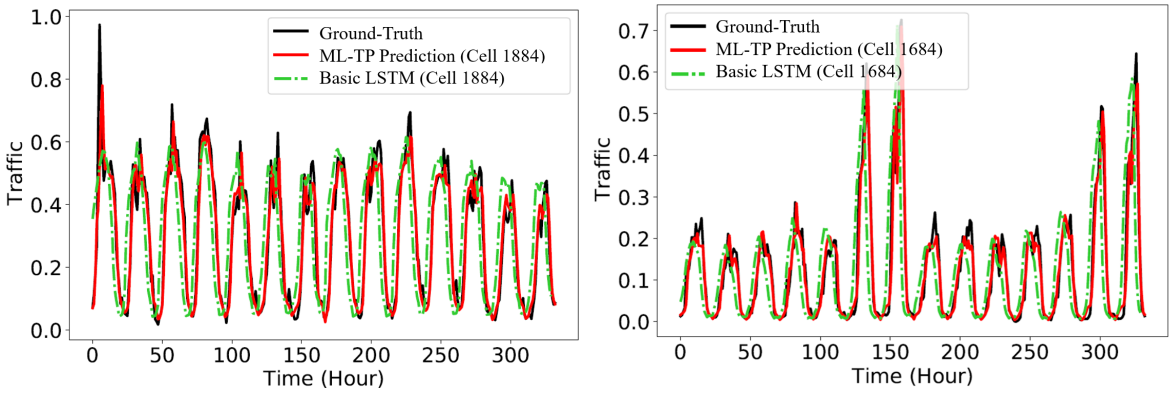


(c)

Fig. 5.9 Performance of ML-TP and the baseline methods in terms of (a) NRMSE; (b) NMAE 2; (c) R2

cell-level mobile traffic patterns. In addition, the conventional LSTM network can mine deep dependencies among mobile traffic loads generated at various time intervals. Therefore, it can be found from Figure 5.9 that the conventional LSTM network outperforms the LR, the ARIMA, and the SVR.

The proposed ML-TP achieves the best results in terms of all the evaluation metrics, i.e., NRMSE, NMAE, and R2. This can be traced back to two main reasons. First, the base-learner for each base-task in ML-TP is based on deep LSTM network and can learn and represent the complex nonlinearities in cell-level mobile traffic load variations. Second, unlike the conventional LSTM network, which is initialised with randomly selected parameter values, the meta-learner in ML-TP will output the best initial searching point in the hypothesis space for a base-learner, leading to a more precise hypothesis for each base-task. Compared with the conventional LSTM network, the meta-learning technique brings about 3.9%, 6.7% and 1.3% performance improvements in terms of NRMSE, NMAE and R2, respectively.



(a) Traffic pattern of cell 1884 (b) Traffic pattern of cell 1684

Fig. 5.10 Predicted mobile traffic load by ML-TP compared the real traffic load of cell (a) 1884 (b) 1684

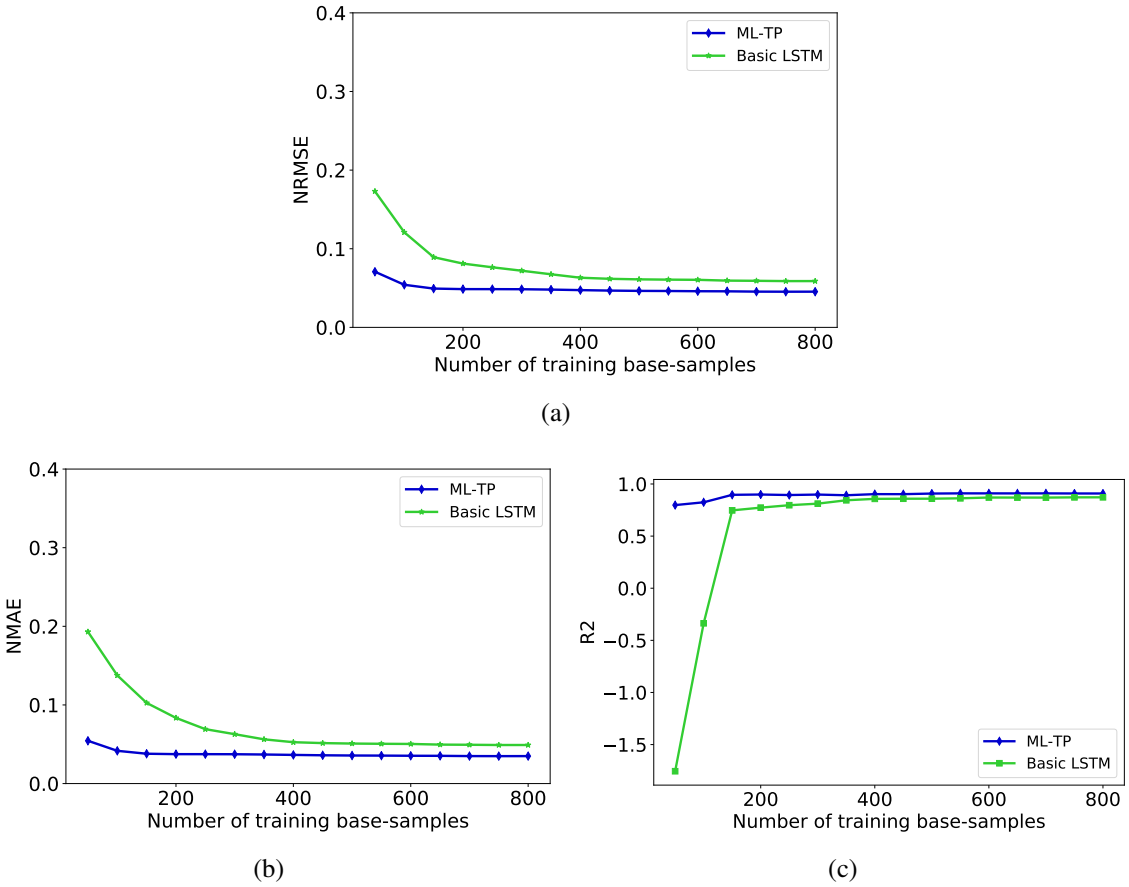
To be more specific, this section also compares the mobile traffic loads predicted by the ML-TP with the ground-truth values. Figure 5.10 shows the predicted mobile traffic loads of ML-TP and the conventional LSTM network, as well as the real mobile traffic loads generated in cells 1884 and 1684, respectively. It can be found that both the conventional LSTM network and the proposed ML-TP can accurately predict the mobile traffic dynamics under normal

traffic patterns. Additionally, ML-TP performs much better than the conventional LSTM network when the mobile traffic patterns have abnormities or sudden changes. This can be explained as the accumulated meta-knowledge will make the base-learners in ML-TP adaptable to abnormal mobile traffic and help them handle unknown traffic patterns. While for the conventional LSTM networks, they will fail to predict unknown mobile traffic variations without meta-knowledge if there are no similar base-samples in their base-task training sets.

5.4.4 Base-samples needed to fine-tune the base-learner of a new base-task

In this subsection, how the meta-learner in ML-TP improves the learning efficiency of the base-learners in terms of the number of training base-samples needed is investigated. Different from the training process in section 5.4.3, for each testing base-task q, a portion of base-samples generated during 11/01/2013-12/15/2013 is randomly selected to fine-tune/train the base-learner initialised by the meta-learner and the conventional LSTM network. Note that for the base-learners in ML-TP and the conventional LSTM networks, the number of training epochs is set to 100, under which those deep learning-based models' parameters have already converged to stable values.

Figure5.11 (a,b,c) describe the evaluation metrics achieved by the ML-TP and the conventional LSTM networks, i.e., without meta-learner, under different numbers of base-samples in each testing base-task's training set. The results are obtained by validating on the testing set. It can be found that the conventional LSTM networks have high NRMSE and NMAE values as well as a low R2 value with small base-task training sets and will obtain a high prediction accuracy if the training set of each testing base-task is large enough (e.g., containing more than 500 training base-samples). This is because with randomly selected initial status, the conventional LSTM networks require abundant training data to adjust their parameter values. Otherwise, they may suffer the overfitting problem. As a comparison, the ML-TP achieves a competitive accuracy performance even with a quite small number of training base-samples for each testing base-task and then will approach stable when that number reaches about



(a)

(b)

(c)

Fig. 5.11 Performance of ML-TP and deep LSTM with respect to the number of training base-samples: (a) NRMSE; (b) NMAE; (c) R2

150. The prediction accuracy of ML-TP with 150 training base-samples exceeds that of the conventional LSTM networks with 800 training base-samples, leading to about 81% reduction in the training base-samples needed. This can be explained as due to the accumulation of meta-knowledge, the base-learner for a new base-task will be given the proper initial values of neuron connection weights and neuronal biases. Thus, it performs well after being fine-tuned with only few training base-samples.

5.4.5 Epochs needed to fine-tune the base-learner of a new prediction task

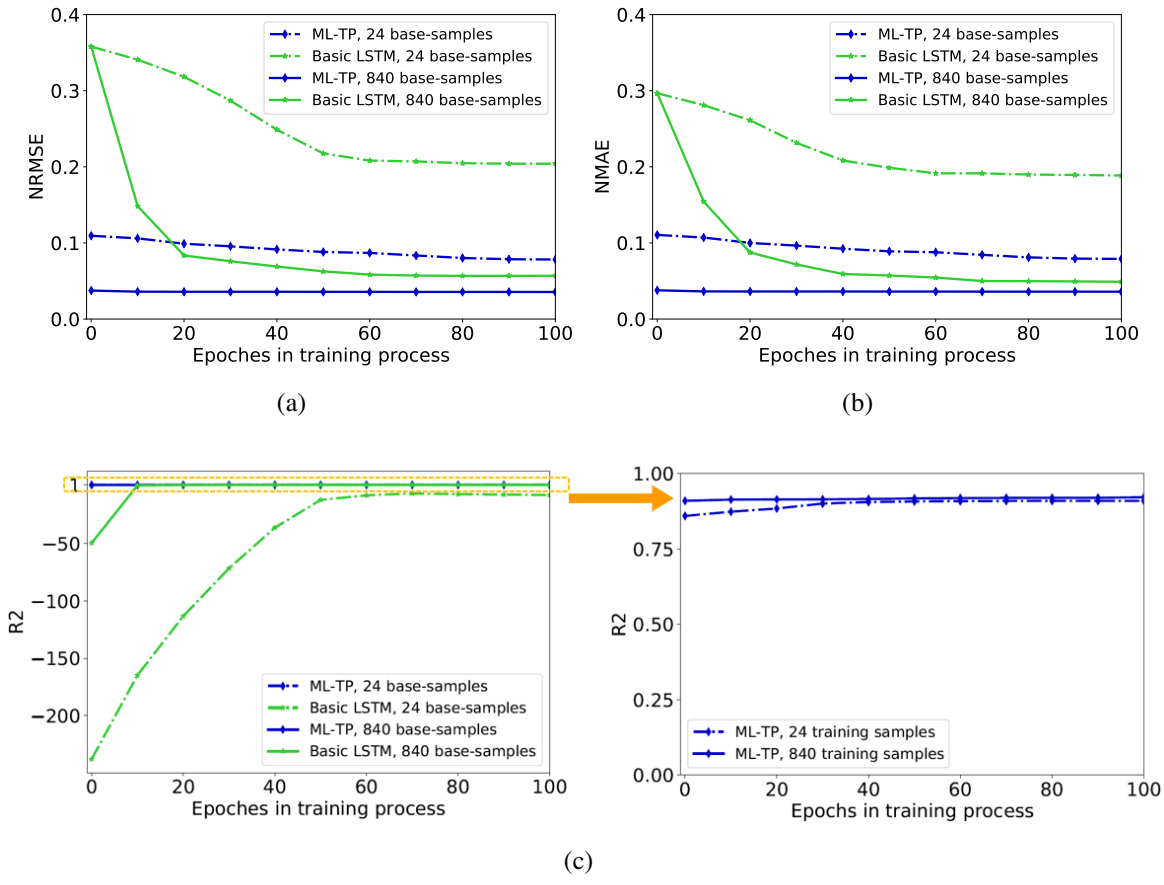


Fig. 5.12 Performance of ML-TP and deep LSTM networks with respect to the number of training epochs: (a) NRMSE; (b) NMAE; (c) R2

This section also examines how the meta-learner in ML-TP improves the learning efficiency of the base-learners in terms of the number of training epochs needed. Figure5.12(a,b,c) show the average NRMSE, NMAE, and R2 achieved by the ML-TP and the conventional LSTM networks over the testing base-tasks versus the number of epochs in the training process, where different number of samples are available. Two scenarios are presented: limited data availability and adequate data availability. One day's data (24 samples) and five weeks' data (840 samples) are provided to construct the training set for each base-task, respectively. The performance of ML-TP and basic LSTM without meta-learner is evaluated under these two scenarios.

It can be seen that for small training sets, the prediction accuracy of the conventional LSTM networks ascends relatively slow as the number of training epochs increases and remains stable at a relatively bad performance after 60 training epochs. This is because the conventional LSTM network will overfit the training set of each testing base-task if the number of base-samples is small. The proposed ML-TP has a faster convergence speed and much better prediction performance than the conventional LSTM networks. This is because due to the accumulation of meta-knowledge, the base-learner for a new base-task in ML-TP will be given the proper initial parameter values that approach the target ones, and thus achieve competitive performance even with few training base-samples and small training epoch numbers. When the base-task training sets have adequate base-samples, the conventional LSTM networks require fewer training epochs to converge and achieve a good prediction accuracy after 40 training epochs. This is in accordance with the previous analysis that the conventional LSTM networks have both high representation ability and training-data dependence due to their complex structures. The proposed ML-TP also has higher predicting accuracy and much faster convergence speed than the conventional LSTM networks when the base-task training sets are large. ML-TP's performance becomes stable when the epoch number exceeds 10, leading to a 75% reduction in the number of training epochs needed compared with the conventional LSTM networks.

5.5 CONCLUSION

In this chapter, an early attempt to introducing the meta-learning concept into cell-level mobile traffic prediction is carried out. A meta-learning-based mobile traffic framework, ML-TP, is propsoed to adaptively learn the proper prediction model for a new mobile cell. In the ML-TP framework, the traffic prediction task for each individual mobile cell is considered as a base learning task and utilise the deep LSTM as the base-learner to address it. This chapter also defines a meta-learning task, aiming to learn to learn the proper prediction models for different base-learning tasks. By testing the ML-TP on a real-world cell-level traffic load dataset, the results reveal that the proposed ML-TP outperforms existing prediction techniques in terms of mobile traffic prediction. Furthermore, compared with the traditional LSTM network, the ML-TP significantly improves the learning efficiency, where the number of training base-samples to fine-tune the base-learner and the requirement on epochs is reduced by about 70% and 80%, respectively.

Except for the initial searching point, whether and how the meta-learning can be adopted to determine the most suitable prediction algorithm and hyper-parameters for a specific mobile traffic prediction task will also be analysed in future work.

Chapter 6

Mobile Network Traffic Prediction Framework: The dmTP

Deep learning technologies have been widely exploited to predict mobile traffic. However, individually training deep learning models for various traffic prediction tasks is not only time consuming but also sometimes unrealistic due to limited traffic records. In this chapter, a novel deep meta-learning based mobile Traffic Prediction framework (dmTP) is proposed, which can adaptively learn to learn the proper prediction model for each distinct prediction task from accumulated meta-knowledge of previous prediction tasks. In dmTP, each mobile traffic prediction task is regarded as a base-task, and an LSTM network is adopted with a fixed structure as the base-learner for each base-task. In order to improve the base-learner's prediction accuracy and learning efficiency, an MLP is further employed as the meta-learner to find the optimal hyper-parameter value and initial training status for the base-learner of a new base-task according to its meta-features. Extensive experiments using real-world datasets demonstrate that while guaranteeing a similar or even better prediction accuracy, meta-learning in the proposed dmTP reduces the number of epoch and base-samples needed to train the base-learners by around 75% 81%, respectively, as compared with the existing prediction models.

6.1 Introduction

With the popularity of mobile devices and applications such as the Internet of things, cloud computing, and virtual reality /augmented reality, mobile communication networks have become an indispensable part of people's lives [169]. According to Cisco's latest statistical report [170], global mobile traffic demands are expected to increase seven-fold from 2016 to 2021 and account for 20% of the total Internet traffic by 2021.

In order to provide mobile service with guaranteed quality [169, 170], key technical challenges in mobile traffic analysis, including mobile traffic prediction, anomaly detection, attack classification, website fingerprinting, and mobile traffic identification, have been widely investigated in recent years [171]. Among these challenges, the accurate prediction of mobile traffic is a critical enabler of advanced management and optimisation of network resources.

Mobile traffic prediction has attracted continued research interest from both academia and industry [162, 172]. Estimating the future loads based on historical traffic records, mobile traffic prediction has been widely considered as a time series forecasting problem. The existing mobile traffic prediction models or algorithms can be generally classified into two categories: statistical methods and machine learning-based methods [52].

In statistical methods, researchers tried to use explicit statistical models with certain parameters to fit the mobile traffic patterns for future traffic forecasting. In [102], Zhou et al. used the ARIMA model to capture the short-term correlation in network traffic. The seasonal ARIMA model was adopted in [85] to capture the long-term traffic correlation. Li et al. [162] demonstrated that the cellular traffic loads generated by three typical mobile services follow heavy-tailed distributions and then utilised the α-stable model to predict their load fluctuations. However, since realistic mobile traffic tends to show complex irregular patterns, it is difficult for statistical models to predict realistic mobile traffic accurately.

Machine learning technologies are gaining increasing popularity in mobile traffic prediction. Unlike the statistical methods, machine learning-based methods package the statistical models used for prediction into opaque or semi-opaque black boxes, which need to be trained using historical traffic records. In [80] and [115], LR and SVR were used to predict the

network-level mobile traffic, respectively. Nevertheless, these shallow learning methods cannot cope with many practical prediction scenarios due to the fact that they cannot perform feature extraction on their own, while relying on some prior knowledge of the input features. Recently, powerful deep learning tools have been leveraged for mobile traffic prediction. Nie et al. [120] employed a deep belief network-based model to predict the mobile traffic loads aggregated across a city. Assuming traffic information of neighbouring cells, Feng et al. [172] proposed an LSTM network-based prediction model to forecast the traffic loads of a target cell. With a reduced connection complexity, a model based on a random connectivity LSTM network was proposed in [123] to predict a single cell's traffic. Furthermore, based on historical traffic loads generated in all the cells, a convolutional neural network-based prediction model [164] and a convolutional LSTM network-based prediction [52] were proposed to forecast the spatial mobile traffic in a city. However, in the existing works, a specific prediction model must be constructed and trained for each individual mobile traffic prediction task as the time series of mobile traffic handled by various prediction tasks are quite different. Separately training the prediction models for multiple tasks is not only time consuming but also unrealistic since there are not always sufficient historical traffic records available, e.g., for newly built networks.

To fill the above gaps, this chapter presents an early attempt to introduce deep meta-learning into mobile traffic prediction and investigate how to make the prediction model learn to learn for a specific mobile traffic prediction task according to its characteristics (meta-features). The main contributions of this work are summarised as follows:

- Through FFT, it can be demonstrated that the temporal variations of mobile traffic over hours, days, and weeks can be characterised by the five main frequency components of the frequency spectrum. More specifically, my analysis shows that the Pearson correlation coefficient of two base-tasks' normalised traffic load series is negatively correlated with the Euclidean distance between their frequency component vectors. In other words, two base-tasks' normalised traffic load series tend to have similar time-domain variations if their frequency component vectors are close to each other, and vice versa.

- This chapter introduces deep meta-learning into mobile traffic prediction, where each prediction task is regarded as a base-task and is represented as a time series forecasting problem. By defining the meta-task as learning to learn the proper prediction models for different base-tasks, a novel deep meta-learning based mobile Traffic Prediction framework (dmTP) is proposed. In dmTP, an LSTM network with a fixed structure is adopted as the base-learner to forecast a base-task's traffic load based on previous values. Using the five main frequency components as the meta-features, an MLP is employed as the meta-learner to output the optimal hyper-parameter value and initial status for the base-learner of a new base-task according to the accumulated meta-knowledge and the base-task's meta-features.

- The performance of dmTP is evaluated by extensive tests using real-world mobile traffic data for heterogeneous prediction tasks. Numerical results show that the meta-learning technology not only improves the prediction models' prediction accuracy and learning efficiency but also makes them more adaptable to high varied traffic patterns.

The rest of this article is organised as follows. Section 6.2 presents the statistical characteristics of mobile traffic and briefly introduces the meta-learning technology. Section 6.3 elaborates on the proposed dmTP. Section 6.4 evaluates the performance of dmTP in comparison with some existing prediction models. In Section 6.5, a summary of this study is given.

6.2 Dataset Description and Background Knowledge of Meta-Learning

6.2.1 Mobile Traffic Traces

This chapter adopts three real-world mobile traffic datasets generated in Milan (Dataset 1), Guangzhou (Dataset 2), and London (Dataset 3), respectively. Table 6.1 provides details

Table 6.1 Description of the three adopted datasets

Dataset 1	**Location**	**Duration**	**Items**
Mobile network traffic	Milan, Italy	11/01/2013-01/01/2014	319,896,289 records
Description: Records are provided by Telecom Italy with a temporal interval of 10 minutes. Milan city area is divided into 9,999 grids, and the size of a grid is about 235m×235m. Each traffic record has information about its time interval, geographical gird, and data volume in bits.			
Dataset 2	**Location**	**Duration**	**Items**
Short message service traffic	Guangzhou, China	03/01/2019-03/31/2019	1,521,005 records
Description: Records provided by China Unicom. Each short message service record has information about its timestamp and data volume.			
Dataset 3	**Location**	**Duration**	**Items**
Twitter	London, UK	02/15/2016-02/28/2016	136,710 records
Description: Twitter records purchased from the Twitter Company. Each record has information about its timestamp and geolocation.			

about those datasets. Dataset 1 is publicly available [52], while Datasets 2 and 3 were purchased and are not publicly accessible.

Each grid in Dataset 1 is referred to as a cell, and the forecasting problem for each cell's mobile traffic load (in bits) is regarded as an individual prediction task. As the city area of Milan is divided into 9,999 grids, there are 9,999 prediction tasks for Dataset 1. Further details about Dataset 1 can be referred to as section 5.2.1. Datasets 2 and 3 aims for the forecasting problem for short message service traffic load (in bits) generated in the whole Guangzhou city and that for Twitter traffic load (in amounts) generated in the whole London city as two prediction tasks, respectively. These prediction tasks from Datasets 1, 2, and 3 focus on different spatial scales and involve different kinds of mobile traffic. This study does not breach user privacy or raises ethical or legal issues. Indeed, this study does not process individual or personal data. Also, the datasets are strongly anonymised by the geographical aggregation at the cellular or city level, ensuring that the mobile demands are merged over numerous subscribers.

6.2.2 Characteristics of Mobile Traffic

The time interval resolution is set to one hour following the settings in [52] mainly because time interval resolutions smaller than one hour will make many cells from Dataset 1 have lots of zero values in their traffic load series, which will be too sparse for traffic prediction.

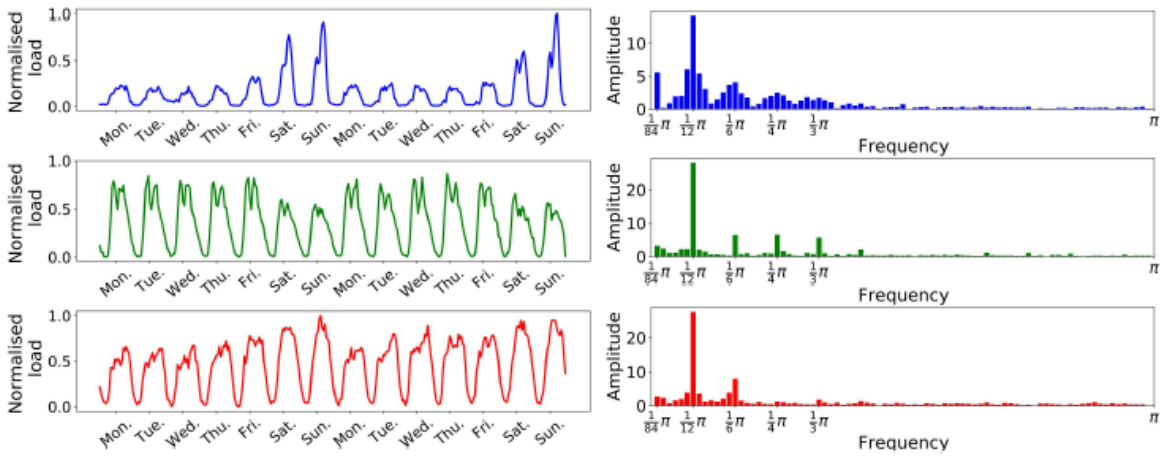


Fig. 6.1 Characteristics of mobile traffic in the time domain (left) and frequency domain (right)

The peak traffic loads among various prediction tasks from the three datasets differ by up to 5 orders of magnitude, and the traffic load series of each prediction task is normalised into the range of [0,1] using the min-max normalisation scaling method, as defined in 3.13. The left of Figure 6.1 illustrates the normalised mobile traffic loads in three prediction tasks for two weeks. It can be observed that though the three mobile traffic streams exhibit different temporal patterns, they all exhibit a weekly cyclic pattern, which is also observed in the time series of other cells from Dataset 1.

6.2.3 Characteristics of Mobile Traffic

A discrete periodic signal is generated for each mobile traffic prediction task by periodically repeating its normalised real traffic loads in the first secular week (from Monday to Sunday), and then FFT is performed. Notably, though using a periodic signal to represent the actual mobile traffic stream may lose some variation information in the time series, it allows depicting the features of a prediction task with much fewer records of historical traffic load. The right of Figure 6.1 shows the amplitudes of the frequency components in the FFT results related to the three time series in the left of Figure 6.1. From The right of Figure 6.1, it can be found that in the frequency domain $\omega = \frac{\pi}{84}, \frac{\pi}{12}, \frac{\pi}{6}, \frac{\pi}{4}, \frac{\pi}{3}$ (corresponding to the periods of one week, one day, 12 hours, 8 hours, and 6 hours, respectively) are the five main

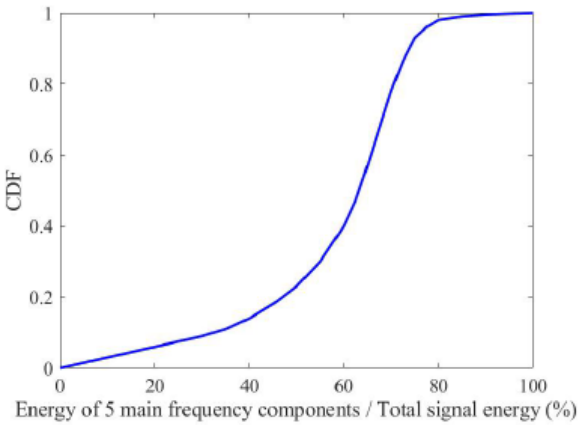


Fig. 6.2 Cumulative distribution function of base-tasks whose sum energy of the five main frequency components accounts a certain percentage of the related periodic signal's energy

frequency components for a mobile traffic prediction task. However, the amplitudes of the main frequency components vary evidently across the different prediction tasks. Figure 6.2 shows the cumulative distribution function of the percentage of the traffic load series' signal energy carried by the five main frequency components. It can be observed that the sum energy of the five main frequency components in more than 60% of the considered prediction tasks exceeds 60% of the signal energy.

A frequency component vector of size ten is used to record the real parts and imaginary parts of a prediction task's main frequency components, same as in section 5.2.2. Obviously, this vector can reflect both the amplitude and the phase of each main frequency component. Figure 5.5 illustrates the inter-dependencies of 5,000 randomly selected pairs of time series (i.e., prediction tasks) from Dataset 1 in both the time domain and the corresponding frequency domain. Specifically, Figure 5.5 plots the Euclidean distance between the two frequency component vectors of a pair of prediction tasks versus the Pearson correlation coefficient between the two corresponding time series. It can be observed that the Pearson correlation coefficient of the two prediction tasks' time series is negatively correlated with the Euclidean distance between their frequency component vectors. In other words, two prediction tasks' time series tend to have similar time-domain variations if their frequency component vectors are close to each other, and vice versa. This implies that the frequency component vector can be used to characterise the features of a prediction task's traffic pattern.

6.2.4 Meta-Learning

Meta-learning studies how learning systems can increase learning efficiency through experience. The goal of meta-learning is to understand how learning itself can become flexible according to the domains or tasks under study [162].

For a typical supervised learning task ξ and a learner ς, each sample is denoted by labelling a number of features with an unknown target function F_ξ and the hypothesis space of learner ς, H_ξ, is defined as the set of all the possible hypothesis functions generated by ς. The training progress of ς can thus be seen as searching the hypothesis function $H_\varsigma(\xi)$ that approximates F_ξ over ς's hypothesis space. ς usually embeds a set of biases, which may be caused by the adopted learning algorithm, hyper-parameters, or the initial status. These biases may restrict the size of a base-learner's hypothesis space, and will affect how the base-learner searches the hypothesis space. Meta-learning matches the biases of a base-learner to an individual task, which is achieved by a meta-task that adaptively generates a proper set of biases for each learning task according to the learning task's meta-features. The meta-task itself can be seen as a learning task and handled by a meta-learner. Accordingly, those original individual learning tasks are referred to as base-tasks.

6.3 The Proposed dmTP

Figure 6.3 shows the diagram of my proposed deep meta-learning based mobile traffic prediction framework, dmTP. In dmTP, each individual mobile traffic prediction task is regarded as a base-task and present an LSTM network-based prediction model with a fixed structure as the base-learner for it. A base-task's frequency component vector is defined as its meta-features. The number of steps, which is a hyper-parameter defining the length of the input sequence and is denoted by TS, and the initial values of neural connection weights and neural thresholds as a base-learner's set of biases. For each considered value of TS, the hypothesis space of a base-learner is constituted by the hypothesis functions that each maps the sequence of TS input vectors to an output value. Thus, the value of TS determines a base-learner's hypothesis space, and the initial parameter values determine the base-learner's

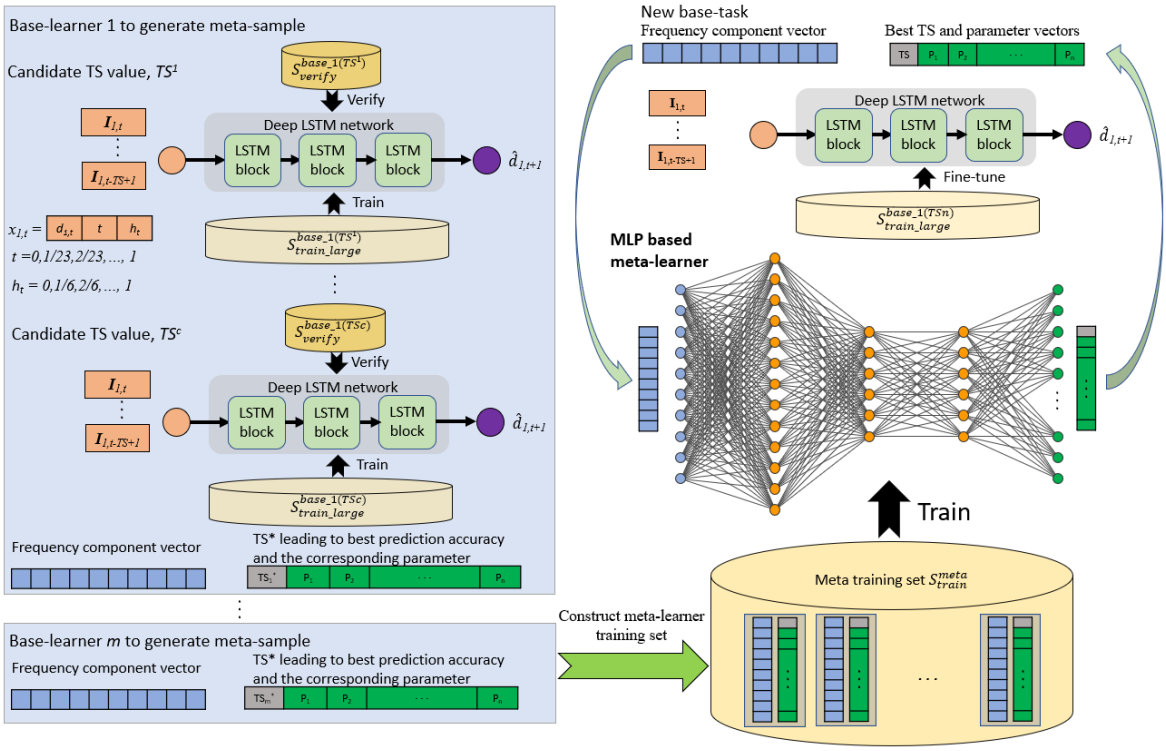


Fig. 6.3 Architecture of the proposed dmTP framework

initial searching point in its hypothesis space. Intuitively, since the meta-features of a base-task reflect its traffic pattern characteristics, the best set of biases of the base-learner will be influenced by this task's meta-features. An MLP is used as the meta-learner to non-explicitly extract the correlation between the base-tasks' meta-features and their best sets of biases, and output the best set of biases for a new base-task's base-learner according to the meta-features.

Notations in dmTP: $\mathbb{S}_{train}^{meta}$ denotes the meta-task training set, which is equivalent to the set of base-tasks generating meta-samples for meta-task training. For base-task m in $\mathbb{S}_{train}^{meta}$, $\mathbb{S}_{train\_large}^{meta\_m(TS)}$ is used to denote the large base-task training set of base-samples for training m's base-learner when the number of steps equals TS, while $\mathbb{S}_{verify}^{meta\_m(TS)}$ is used to denote the set of base-samples for verifying the base-learner's performance. For base-task n not in $\mathbb{S}_{train}^{meta}$, $\mathbb{S}_{train\_small}^{meta\_n(TS_n^*)}$ is used to denote the small base-task training set with few base-samples for fine-tuning the base-learner and $\mathbb{S}_{test}^{meta\_n(TS_n^*)}$ is used to denote the set of base-samples for testing the base-learner's performance.

6.3.1 Deep LSTM Network as the Base-learner

In dmTP, a multi-layer LSTM network with L layers is constructed to act as the base-learner. For a specific prediction task, this LSTM network will be continuously fed a sequence of input vectors related to previous TS time intervals and predict (as the output) normalised mobile traffic load in the next time interval. Each input vector consists of three attributes: the normalised mobile traffic load, the day of the week, and the hour of the day.

The structure of an LSTM memory has been shown in Figure 3.3, and as stated in Chapter 5.3.2, the number of parameters to be trained in each layer is given by $4 \times (U_m + V_m) \times V_m + 4 \times V_m$ and the total number of parameters to be trained in an LSTM network-based base-learner is given in equation 5.9.

6.3.2 Train the Meta-learner with Meta-samples

It is assumed that a base-learner will obtain the highest prediction accuracy and training efficiency when its TS value determines the proper hypothesis space and its initial parameters approach the target ones (the initial searching point in the hypothesis space approaches the target function) [162].

The dmTP utilises a set of base-tasks to construct $\mathbb{S}_{train}^{meta}$. For each base-task m in $\mathbb{S}_{train}^{meta}$, the best value of TS for its base-learner is selected from multiple candidates through exhaustive trials. Specifically, for every candidate value, TS^c, the base-learner is trained using $\mathbb{S}_{train\_large}^{base\_m(TS^c)}$ with randomly selected initial values of the PN parameters, and its performance is verified via $\mathbb{S}_{verify}^{base\_m(TS^c)}$. All the base-samples in $\mathbb{S}_{train\_large}^{base\_m(TS^c)}$ and $\mathbb{S}_{verify}^{base\_m(TS^c)}$ have the same length as the input sequence, TS^c. The best TS value that leads to the base-learner's highest prediction accuracy is then selected and denoted by TS_m^*. By labelling base-task m's frequency component vector with TS_m^* and the PN parameters of the base-learner trained by $\mathbb{S}_{train\_large}^{base\_m(TS_m^*)}$, one meta-sample is obtained.

The meta-learner with MLP is constructed which consists of at least three layers of operations [45]. For the meta-learner, the input is the frequency component vector, and the output is the predicted PN parameters and the predicted optimal TS. The MLP based meta-

learner is trained using $\mathbb{S}_{train}^{meta}$. With the MLP's ability of feature extraction and correlation characterization [45], the meta-learner will be able to generate the best step number, TS_m^*, and initial parameter values approaching the target ones, for a given frequency component vector as its input, for the base-learner of a new base-task n.

6.3.3 Fine-tune the Base-learner for a New Base-Task

As shown in Figure 6.3, for a new mobile traffic prediction task (i.e., a new base-task) n, the frequency component vector is first extracted as its meta-features. The well-trained meta-learner is fed with the meta-features and outputs a set of biases for the base-learner of base-task n. Specifically, for a new mobile traffic prediction task, the well-trained meta-learner takes the task's traffic features as its input, and aims to give the optimal initial parameters and the hyperparameter, TS, to construct a base-learner for the base-task.

The base-learner will take the given TS_n^* as its number of steps and set its initial values of neural connection weights and neural thresholds according to the output of the meta-learner. Then, the base-learner is fine-tuned using the small base-task training set, $\mathbb{S}_{train\_small}^{base\_n(TS_n^*)}$. With meta-knowledge, the base-learner of n is expected to obtain a good prediction accuracy and training efficiency in terms of a fast convergence speed and a small number of base-samples needed. Guaranteeing the base-learner's prediction accuracy, $\mathbb{S}_{train\_small}^{base\_n(TS_n^*)}$ can only contain a few base-samples.

6.4 Evaluation on Real-world Mobile Traffic Data

6.4.1 Experimental Settings

In the experiments, the base-learner is constructed as a three-layer LSTM network, where the output vectors of the first and the second LSTM memory blocks are both of size 5. According to the LSTM memory block structure and equation 5.9, the total number of parameters to be trained for each base-learner is 428. The meta-learner has three hidden layers of size 300, 300, and 400, respectively, while its input and output layers have ten neurons and 429

neurons (one for TS and the rest for the 428 parameters in a base-learner to be trained), respectively.

The experiment randomly selects 8,000 out of 9,999 base-tasks related to Dataset 1 to construct the meta-training set, $\mathbb{S}_{train}^{meta}$, to train the meta-learner. For each base-task m in $\mathbb{S}_{train}^{meta}$, TS is tested whose value ranges from 3 to 24. For a candidate TS value, TS^c, a sliding window with size TS^c, is applied to split m's normalised traffic load series and generate the base-samples by labelling each sequence of m input vectors with the normalised traffic load in the next time interval. 90% of these base-samples are selected to construct the base-task's training set, i.e., $\mathbb{S}_{train\_large}^{base\_m(TS^c)}$ while use the other ones to construct the test set, i.e., $\mathbb{S}_{verify}^{base\_m(TS^c)}$. The performance of dmTP is examined on the remaining 1,999 base-tasks of Dataset 1 that have not been selected for meta-training and the two testing base-tasks from Datasets 2 and 3. Similarly, for each testing base-task n, a sliding window with size TS_n^* is applied, which is output by the meta-learner, to generate the base-samples. For each testing base-task n, the traffic load series are also divided into a training set and a test set, where the training set is used to fine-tune the base-learner, and the test set is used to evaluate the performance of the base-learner.

The performance of dmTP is compared with the existing time series forecasting methods, including ARIMA [102], LR [80], SVR [115], and basic LSTM networks [123] for $TS = 12, 24$. For a fair comparison, a basic LSTM network with the same structure as a base-learner is constructed in dmTP. the MSE loss is chosen as the loss function. At the same time, the adaptive moment estimation algorithm [45] with the default learning rate is utilised to optimise the baseline LSTM networks as well as the meta-learner and base-learners in dmTP. It can be noted that for around 20% of the base-tasks in Dataset 1, the traffic loads in more than 50% of time intervals are less than 20% of the peak traffic load, while for around 84% of the base-tasks in Dataset 1 and the two base-tasks from Datasets 2 and 3, the traffic loads in more than 50% of time intervals are less than 40% of the peak traffic load. Considering that the mean values of the traffic load series related to a considerable portion of base-tasks are relatively low, the NRMSE is used to evaluate the accuracy of the considered prediction methods.

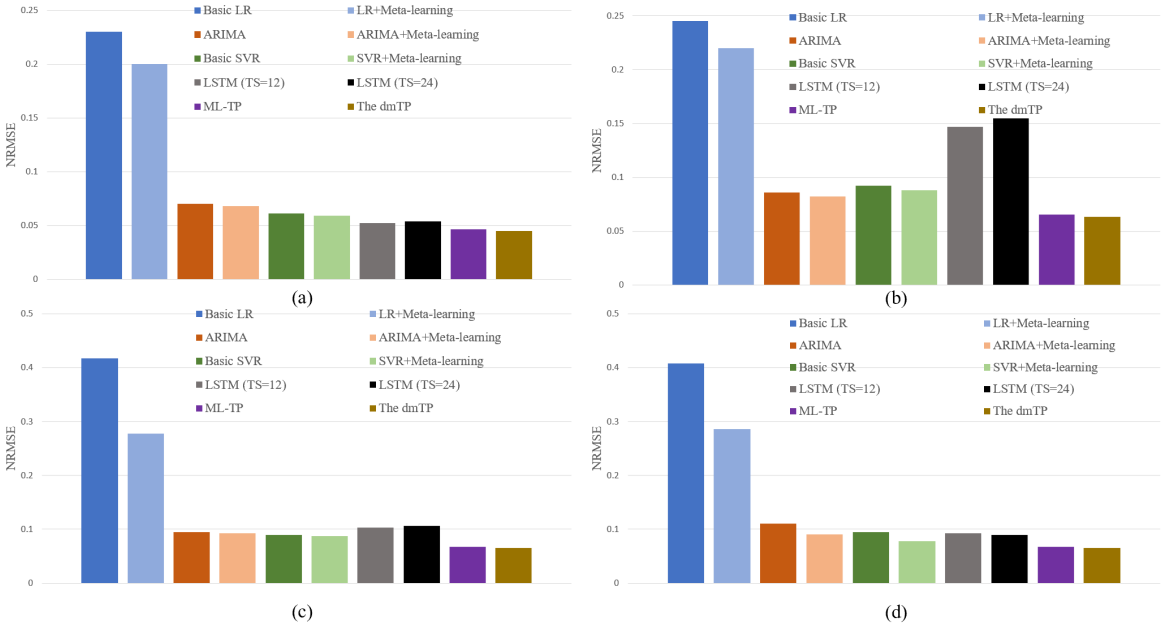


Fig. 6.4 Performance of dmTP and the baseline methods

6.4.2 Prediction Performance

Figure 6.4 compares the prediction accuracy achieved by dmTP, ML-TP and the baseline methods. In order to test the applicability of meta-learning technology to other prediction methods, the performance of ARIMA, LR, and SVR is evaluated when their hyper-parameters (i.e., the number of steps for ARIMA and LR, and the kernel function for SVR for a certain base-task are either fixed or selected by MLP based meta-learners from multiple candidate values (where candidate TS value ranges from 1 to 5 for ARIMA, candidate TS value ranges from 1 to 10 for LR, and candidate kernel functions for SVR are linear, poly, and exponential functions). Except for the output layer, the meta-learners for the above three baseline methods have the same structure as the meta-learner in dmTP. For each testing base-task from Dataset 1, the testing base-samples are generated during the two weeks of 12/16/2013-12/22/2013 and 12/23/2013-12/29/2013, where the traffic pattern during the second week has apparent variations due to the Christmas holidays. For the two testing base-tasks from Datasets 2 and 3, the testing base-samples are generated in the last week, i.e., 03/25/2019-03/31/2019 and 02/22/2016-02/28/2016, respectively. Note that for each testing base-task in Figure

6.4, all the base-samples that have not been used for testing are used to fine-tune/train the base-learner in dmTP and the baseline models.

As can be seen from Figure 6.4, ARIMA and LR perform the worst among all the considered methods. This is because these simple models are not able to capture the highly nonlinear temporal patterns of mobile traffic loads. The SVR, a nonlinear prediction method, can deal with the nonlinearities in load variation and thus achieve better performance than ARIMA and LR. Due to the deep learning capability, the basic LSTM networks with adequate training data can learn the deep dependency between traffic loads generated in various time intervals and thus perform better than ARIMA, LR, and SVR, as shown in Figure 6.4 (a). However, Figure 6.4 (c) (d) show that the prediction accuracy of the basic LSTM networks degrades for prediction tasks related to Datasets 2 and 3. This is because the basic LSTM networks require a large number of samples to train their models. The small training sets with base-samples generated in a three-week or one-week period will lead to overfitting and thus poor performance. Figure 6.4 (b) shows that when there are high variations in the testing traffic patterns, the basic LSTM networks have unsatisfactory performance. This is because these basic LSTM networks have a high dependency on the training data, and they will fail to predict the highly varied traffic loads, which are apparently different to usual patterns if there are not similar samples in their training sets. The prediction performance achieved by ML-TP proposed in the last chapter is slightly inferior to prediction performance than the dmTP. This is because the hyper-parameters are fixed in ML-TP. It cannot determine the hyper-parameters, which also have apparent impacts on the prediction performance.

The proposed dmTP always obtains the best prediction accuracy, attributing to two aspects. First, the base-learner for each base-task in dmTP has the ability to learn and representing the complex nonlinearities in mobile traffic load variations. Second, unlike the basic LSTM networks embedded with fixed hyper-parameters and randomly selected initial parameter values, the meta-learner in dmTP will find the best TS and proper initial values of parameters for a base-learner, which leads to higher prediction accuracy and stronger adaptability to the diverse traffic patterns. Compared with ARIMA, LR, and SVR, dmTP reduces the NRMSE by about 25-43%, 73-85%, and 20-39%, respectively, for the testing

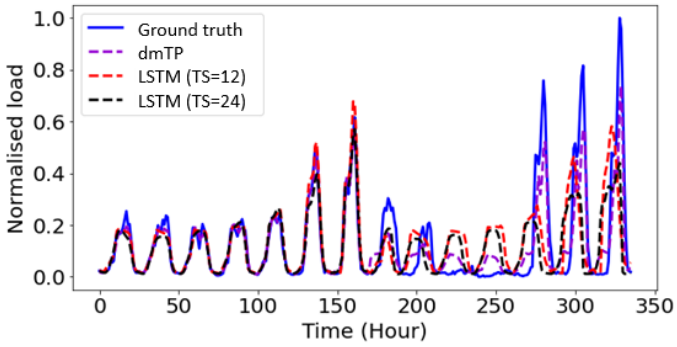


Fig. 6.5 Prediction results of the dmTP and basic LSTM networks for a testing base-task in Dataset 1 (cell 1684)

base-tasks from Datasets 1, 2, and 3. As shown in Figure 6.4 (a), when there is adequate training data, and the traffic pattern is typical, dmTP can further reduce the NRMSE by 12% compared with the basic LSTM networks, mainly due to the selection of the optimal TS value. As shown in Figure 6.4(b)-(d), when the training sets are small, or the traffic pattern is quite different than usual, dmTP outperforms the basic LSTM networks with a 28-60% reduction in NRMSE owing to the proper selection of initial parameter values in each base-learner. Figure 6.5 shows the prediction results of the dmTP and the basic LSTM networks for a testing base-task in Dataset 1 (cell 1684). It can be clearly seen that the dmTP achieves more accurate prediction values than the basic LSTM networks when the traffic pattern has abnormities or sudden changes. From Figure 6.4, it can also be found that the three baseline methods in conjunction with meta-learning technology perform better than their counterparts without meta-learning technology. The meta-learners can improve the prediction accuracy by about 9%, 17%, and 8%, for ARIMA, LR, and SVR, respectively. These results verify the applicability of the meta-learning technology to ARIMA, LR, and SVR.

Figure 6.6 shows the Average Training Time (ATT) needed by the base-learners in dmTP and the baseline methods for the testing base-tasks as well as the Time needed to Construct the Meta-task training set and Train the Meta-learner (TCMTM). The dmTP consumes more TCMTM than ML-TP, since dmTP needs to train multiple base-learners with different

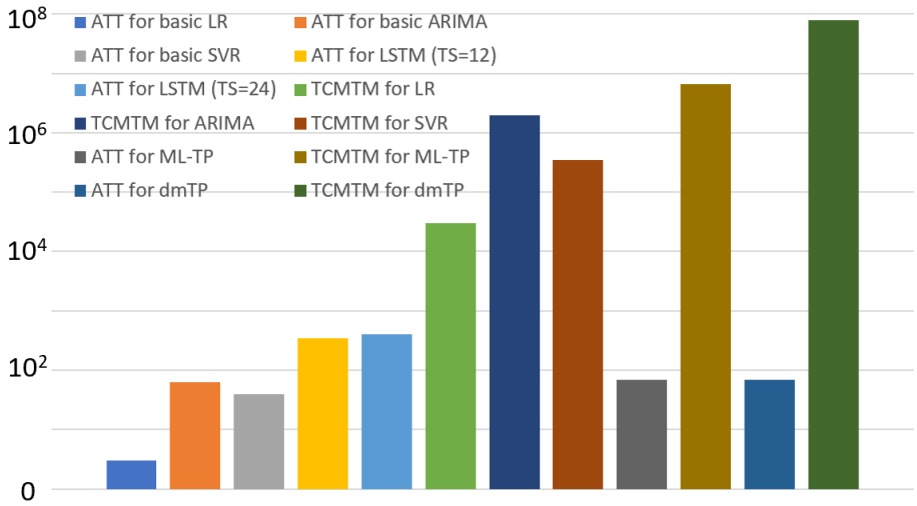


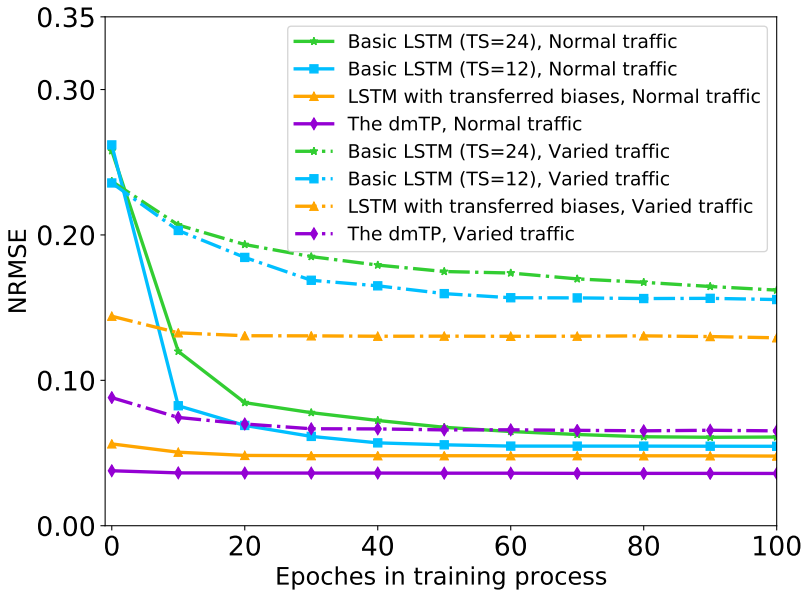
Fig. 6.6 ATT and TCMTM needed by dmTP and the baseline methods

hyperparameters for each base task. Compared with dmTP, the hyper-parameters are fixed such that ML-TP only needs to train one base-learner for each base-task.

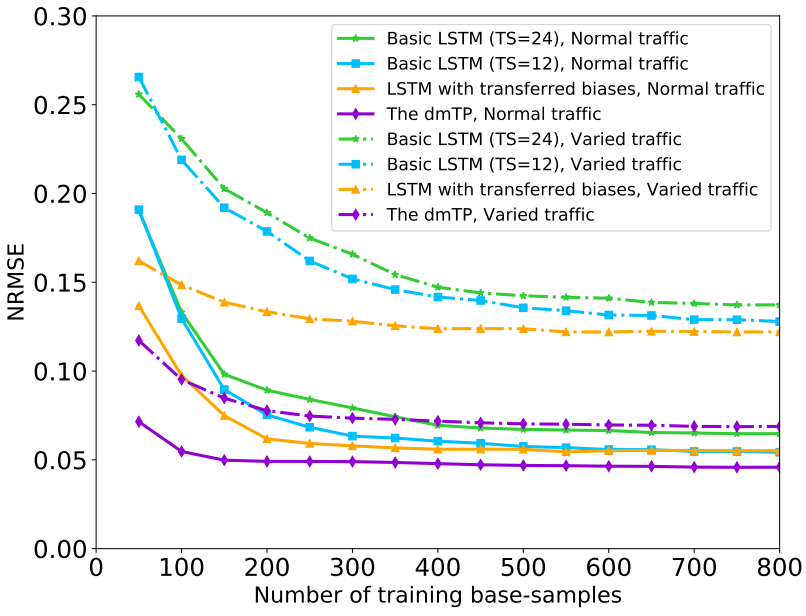
Compared with the baseline methods, although dmTP consumes much more TCMTM, after the initial parameter values have been properly set, the base-learner of a new base-task will converge faster and need less ATT than the basic LSTM networks. Note that since the meta-samples can be obtained off-line and the meta-learner only needs to be trained once, the extra off-line complexity caused by meta-learning in dmTP can be justified by the improved online convergence speed and reduced training time of base-learners.

6.4.3 Learning Efficiency Improvement of Base-learner

In Figure 6.7, the testing base-tasks from Dataset 1 is used to further examine how the meta-learner can help the base-learners improve their learning efficiency in terms of convergence speed and the number of base-samples needed. For each testing base-task, the testing base-samples are generated during the two weeks of 12/16/2013-12/22/2013 and 12/23/2013-12/29/2013, while different from in Figure 6.4, only a portion of the remaining base-samples are randomly selected to fine-tune/train the base-learner in dmTP and the baseline models. Figure 6.7 (a) shows the average NRMSE achieved by the proposed dmTP, basic LSTM networks with random initial parameters, and the LSTM network with biases transferred



(a)



(b)

Fig. 6.7 Performance of dmTP and basic LSTM networks under different numbers of training epochs and different numbers of base-samples

from a certain base-task in $\mathbb{S}_{train}^{meta}$ (cell 6395 with the best TS of 6) versus the number of epochs in training, where 840 base-samples are used to fine-tune/train the base-learner in dmTP and the baseline models for each testing base-task. It can be found that for predicting normal mobile traffic, the NRMSE of basic LSTM networks decreases as the number of epochs increases and remains at a good accuracy level after 40 epochs. Due to the diversity of mobile traffic patterns among various base-tasks, the LSTM network with transferred biases has a slower convergence speed and a higher NRMSE than dmTP. However, the transferred initial parameters lead to a lower initial NRMSE value and a higher convergence speed than the basic LSTM networks. The dmTP's performance becomes stable after ten epochs, leading to a 75% reduction in the number of epochs needed than the basic LSTM networks. Thanks to the chosen optimal TS value, dmTP reduces the NRMSE by about 12% compared with the basic LSTM networks. It can also be seen that dmTP has a much faster convergence speed for predicting highly varied mobile traffic than both the basic LSTM networks and the LSTM network with transferred biases. Moreover, the accuracy improvement of dmTP over the baselines after they all become stable is much larger than predicting normal mobile traffic. This demonstrates that the meta-learner can not only elevate the base-learners' learning efficiency but also make them more robust and adaptable to diverse mobile traffic because the accumulated meta-knowledge will help each base-learner in dmTP handle unknown traffic patterns.

Figure 6.7 (b) displays the average NRMSE achieved by dmTP and the baselines versus the number of training base-samples selected to fine-tune/train the predicting models for each testing base-task with 100 training epochs. It can be seen that for predicting normal mobile traffic, the basic LSTM networks and the LSTM network with transferred biases obtain a high prediction accuracy if the base-task training sets are large enough (e.g., more than 500 training base-samples). The prediction accuracy of the proposed dmTP with 150 training base-samples exceeds that of the basic LSTM networks with 800 training base-samples. Compared with the basic LSTM networks, the meta-learner in dmTP helps the base-learners reduce the training base-samples needed by about 81%. Since each base-learner for a testing base-task in dmTP is given a proper set of biases by the meta-learner, only a limited number

of base-samples are required to fine-tune a base-learner to achieve high accuracy. The NRMSE of dmTP is significantly lower for predicting varied traffic than those of the basic LSTM networks and the LSTM network with transferred biases. This is because, without meta-knowledge, the LSTM networks will fail to accurately predict unknown traffic patterns if there are no similar base-samples in their training sets.

6.5 Conclusion

By taking each individual mobile traffic prediction task as a base-task and the main frequency components of traffic time series as a base-task's meta-features, a meta-task has been constructed which can adaptively learn to learn the proper hyper-parameter and initial parameter values based on accumulated meta-knowledge for the prediction model of a new base-task. Specifically, an LSTM network with a fixed structure is adopted as the base-learner for each base-task while proposing an MLP based meta-learner for the meta-task.

The prediction accuracy of the proposed dmTP framework has been tested on multiple real-world mobile traffic datasets. Experimental results demonstrate that the proposed framework can forecast mobile traffic loads for prediction tasks with quite different spatial scales or application types. With meta-learning, the proposed framework can achieve a much higher prediction accuracy than the existing prediction methods. Moreover, the meta-learner will lead to about 75% and 81% reduction in epochs and base-samples needed to fine-tune the base-learners compared with traditional LSTM network-based prediction models.

Building on this work, it will be interesting to mathematically model and analyse the correlation between a base-task's frequency component vector and the best set of biases for the task's base-learner, as well as investigate whether the meta-learning technology can be used to optimise the structure of the base-learner for a prediction task. Additionally, it is worth studying if some other characteristics of mobile traffic can be considered as a base-task's meta-features.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis mainly focuses on predicting mobile traffic, from mobility prediction and mobile network traffic load prediction perspectives, based on deep-learning techniques. Since mobile traffic has experienced explosive growth in the past decades, mobile traffic analysis becomes an important topic attracting more attention. In the meantime, challenges are also brought for predicting mobile traffic. Firstly, many factors affect mobile traffic. It is difficult to build a mathematical model to model traffic dynamics accurately. Although some methods have been proposed to address this kind of time series prediction tasks, such as ARIMA, the prediction accuracy is still not satisfactory. Moreover, how to select and extract feature also contributes to the prediction performance. Finally, although some machine learning-based approaches are proposed, and the deep learning-based methods can achieve competitive prediction accuracy, they have high computation complexity and require a large number of samples to train the model.

Mobile traffic analysis consists of three main perspectives: mobility analysis, network analysis, and social analysis. This thesis mainly focuses on mobility prediction and network traffic prediction from the first two perspectives, respectively, based on machine learning techniques. Chapter 3 focuses on mobility prediction in mobile networks. The user movement shows temporal correlation and regularity. To address this problem, the LSTM network is

introduced. As an elegant machine learning technique, LSTM shows competitive performance in dealing with time series prediction. First, data pre-processing techniques are applied to the original user traces to filter out the abnormalities and convert them to traces with fixed sampling frequency. A POI extraction scheme is then proposed to extract the areas where the user stays for a long time. After that, the features of the POIs are extracted as the input features to the LSTM network. Finally, an LSTM network is trained and evaluated. The results show that, compared with traditional neural networks, ANN, the prediction accuracy of LSTM achieves 79.7%, which is improved by 45.2% compared with ANN.

For the network analysis, this thesis focuses on network traffic prediction from two perspectives: traffic feature analysis and traffic load prediction. Traffic feature analysis plays a vital role in helping to understand traffic patterns and traffic load prediction. Chapter 4 firstly studies the temporal characteristics of online social networks, which has not been thoroughly analysed in previous research. Through analysis, the Twitter traffic pattern shows periodicity and high similarity on a daily basis. At the same time, it also has a high deviation from day to day. Based on this feature, further statistical analysis is conducted. A data filtering scheme is then proposed to extract regularity features and filter out irregular components in Twitter traffic. The filtered Twitter traffic obtained from the statistical analysis and data filtering scheme is then used to train an LR model. By evaluating the LR model with real Twitter traffic, the proposed LR model achieves higher Twitter traffic prediction accuracy compared with the LR model without statistical analysis and data filtering. This chapter shows the importance of feature analysis, which can improve prediction performance while keeping low computing complexity.

Targeting the mobile network traffic prediction, conventional works using mathematical models or shallow machine learning techniques cannot achieve satisfactory prediction accuracy. Later works utilising deep learning algorithms improve the prediction accuracy at the cost of high computing complexity, more significant data requirements, and longer modelling time. To address these problems, a meta-learning based mobile network traffic prediction framework is proposed. By considering each traffic prediction task as a base-task, a meta-learner is built to learn to learn the proper prediction models from base-tasks. By

testing the framework on a real-world cellular network traffic load dataset, the results reveal that the proposed MLCTPF outperforms existing prediction techniques in terms of traffic load prediction accuracy. In addition, compared with the traditional LSTM network, the MLCTPF significantly improves the learning efficiency by reducing the training epochs and number of training samples needed.

Based on the results achieved in Chapter 5, an advanced mobile network traffic prediction framework, dmTP, is further proposed, which employs MLP as the meta-learner instead of KNN-based meta-learner. Compared with the MLCTPF, the dmTP not only predicts the initial status for the new base-learner but also determine the optimal hyper-parameter. By testing with two datasets with a limited number of samples, a Twitter traffic dataset and a traffic load dataset in another city with different spatial scales, the dmTP can also achieve the best prediction performance. Baseline methods are also integrated with meta-learning technology, and the results show that the baseline methods perform better than their counterparts without meta-learning technology. Furthermore, the dmTP in predicting varied mobile network traffic load is investigated, and results show that the dmTP can improve the prediction accuracy. Finally, the dmTP improves learning efficiency than the traditional LSTM network by reducing the epochs and number of base-samples needed for fine-tuning.

7.2 Future Work

This thesis conducts mobility analysis and traffic load analysis in mobile networks. For mobility analysis, a POI extraction scheme is proposed, and then the deep learning algorithm is used to predict user mobility. In terms of traffic load analysis, statistical analysis and data filtering strategies are proposed to extract the traffic features. Then a novel deep learning-based traffic load prediction framework is established to predict mobile network traffic. The traffic prediction accuracy and learning efficiency of the proposed framework have been evaluated. There are still some potential research directions, which are summarised as follows.

In Chapter 3, user mobility is predicted by the LSTM using five features extracted from the user trajectory. In reality, the user movement is affected by many factors, such as weather, public holiday, and even the user's individual information such as age and job. In the future, how to extract more features and evaluate their effects on user trajectory should be investigated. In addition, these new features should be integrated into the prediction models to improve prediction accuracy.

In Chapter 4, the feature analysis of the mobile traffic load is presented. Statistical analysis is conducted to study the temporal characteristics of Twitter traffic. Based on the statistical analysis, a data filtering scheme is proposed to extract the regularity and filter out the outliers of Twitter traffic. An LR model is established to predict the Twitter traffic load. In the future, the versatility, scalability, and feasibility of the model should be further investigated, for example, the model performance on other OSNs, and the feasibility of using OSN as a proxy to predict the cellular network traffic.

In Chapter 5, a deep learning-based mobile network traffic prediction framework is proposed to predict the cellular network traffic load. The prediction accuracy and learning efficiency of the proposed framework are evaluated. However, the current framework only predicts the model parameters of a fixed type of model with a fixed structure, i.e., a deep LSTM network with a fixed network structure. In the future, the meta-learning concept can be expanded to more types of models with various structures.

Chapter 6 proposes an advanced mobile network traffic prediction framework, dmTP, based on the results achieved in Chapter 5. The meta-learner is enabled to predict both the hyper-parameter and initial status for the base-learner. In the future, the meta-learner can be empowered such that the meta-learner can determine not only the hyper-parameters and initial status for the base-learner, but also can decide the optimal model and its structure for the base-learner. It is also worth exploring other mobile network traffic features as the base-task's meta-features to improve prediction performance further.

References

[1] D. Naboulsi, M. Fiore, S. Ribot, and R. Stanica, "Large-Scale Mobile Traffic Analysis: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 124–161, 2016.

[2] A. A. Nanavati, S. Gurumurthy, G. Das, D. Chakraborty, K. Dasgupta, S. Mukherjea, and A. Joshi, "On the structural properties of massive telecom call graphs: findings and implications," in *Proceedings of the 15th ACM international conference on Information and knowledge management*, pp. 435–444, 2006.

[3] GSMA, "Unique mobile subscribers," 2021.

[4] P. R. Center, "Emerging nations embrace Internet, mobile technology," 2014.

[5] D. Doran, V. Mendiratta, C. Phadke, and H. Uzunalioglu, "The importance of outlier relationships in mobile call graphs," in *2012 11th International Conference on Machine Learning and Applications*, vol. 2, pp. 24–29, IEEE, 2012.

[6] S. Yang, B. Wu, and B. Wang, "Multidimensional views on mobile call network," *Frontiers of Computer Science in China*, vol. 3, no. 3, pp. 335–346, 2009.

[7] V. Soto, V. Frias-Martinez, J. Virseda, and E. Frias-Martinez, "Prediction of socioeconomic levels using cell phone records," in *International Conference on User Modeling, Adaptation, and Personalization*, pp. 377–388, Springer, 2011.

[8] A. Wesolowski, N. Eagle, A. J. Tatem, D. L. Smith, A. M. Noor, R. W. Snow, and C. O. Buckee, "Quantifying the impact of human mobility on malaria," *Science*, vol. 338, no. 6104, pp. 267–270, 2012.

[9] E. Enns and J. Amuasi, "Human mobility and communication patterns in Cote d'Ivoire: A network perspective for malaria control," *NetMob D4D Challenge*, pp. 1–14, 2013.

[10] J. P. Leidig, Y. Kitsumi, K. A. O'Hearn, C. M. Sauer, J. Scripps, and G. Wolffe, "Applying mobile datasets in computational public health research," in *Proc. NetMob D4D Challenge*, pp. 1–11, 2013.

[11] M. Kafsi, E. Kazemi, L. Maystre, L. Yartseva, M. Grossglauser, and P. Thiran, "Mitigating epidemics through mobile micro-measures," *arXiv preprint arXiv:1307.2084*, 2013.

[12] U. Paul, A. P. Subramanian, M. M. Buddhikot, and S. R. Das, "Understanding traffic dynamics in cellular data networks," in *2011 Proceedings IEEE INFOCOM*, pp. 882–890, IEEE, 2011.

[13] E. Halepovic and C. Williamson, "Characterizing and modeling user mobility in a cellular data network," in *Proceedings of the 2nd ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pp. 71–78, 2005.

[14] X. Lu, E. Wetter, N. Bharti, A. J. Tatem, and L. Bengtsson, "Approaching the limit of predictability in human mobility," *Scientific reports*, vol. 3, p. 2923, 2013.

[15] J. Scourias and T. Kunz, "An activity-based mobility model and location management simulation framework," in *Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pp. 61–68, 1999.

[16] P. A. Gagniuc, *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017.

[17] A. Ulvan, M. Ulvan, and R. Bestak, "The enhancement of handover strategy by mobility prediction in broadband wireless access," in *Proceedings of the networking and electronic commerce research conference (NAEC 2009)*, pp. 266–276, American Telecommunications Systems Management Association Inc., 2009.

[18] S. H. S. Ariffin, N. N. N. Abd, and N. E. Ghazali, "Mobility prediction via Markov model in LTE femtocell," *International Journal of Computer Applications*, vol. 65, no. 18, 2013.

[19] N. A. Amirrudin, S. H. S. Ariffin, N. N. N. Abd Malik, and N. E. Ghazali, "User's mobility history-based mobility prediction in LTE femtocells network," in *2013 IEEE International RF and Microwave Conference (RFM)*, pp. 105–110, IEEE, 2013.

[20] A. Hadachi, O. Batrashev, A. Lind, G. Singer, and E. Vainikko, "Cell phone subscribers mobility prediction using enhanced Markov Chain algorithm," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pp. 1049–1054, IEEE, 2014.

[21] S. Gambs, M.-O. Killijian, and M. N. del Prado Cortez, "Next place prediction using mobility markov chains," in *Proceedings of the first workshop on measurement, privacy, and mobility*, pp. 1–6, 2012.

[22] B. C. Csáji, A. Browet, V. A. Traag, J.-C. Delvenne, E. Huens, P. Van Dooren, Z. Smoreda, and V. D. Blondel, "Exploring the mobility of mobile phone users," *Physica A: statistical mechanics and its applications*, vol. 392, no. 6, pp. 1459–1473, 2013.

[23] F. Simini, M. C. González, A. Maritan, and A.-L. Barabási, "A universal model for mobility and migration patterns," *Nature*, vol. 484, no. 7392, pp. 96–100, 2012.

[24] S. Scepanovic, P. Hui, and A. Yla-Jaaski, "Revealing the pulse of human dynamics in a country from mobile phone data," *NetMob D4D Challenge*, pp. 1–15, 2013.

[25] Y. Yang, C. Herrera, N. Eagle, and M. C. Gonzalez, "A multi-scale multi-cultural study of commuting patterns incorporating digital traces," in *Proc. NetMob*, pp. 1–3, 2013.

[26] A. Hess, I. Marsh, and D. Gillblad, "Exploring communication and mobility behavior of 3G network users and its temporal consistency," in *2015 IEEE international conference on communications (ICC)*, pp. 5916–5921, IEEE, 2015.

[27] M. C. González, C. A. Hidalgo, and A.-L. Barabási, "Understanding individual human mobility patterns," *Nature*, vol. 453, pp. 779–782, jun 2008.

[28] F. Xu, Y. Li, H. Wang, P. Zhang, and D. Jin, "Understanding Mobile Traffic Patterns of Large Scale Cellular Towers in Urban Environment," *IEEE/ACM Transactions on Networking*, vol. 25, pp. 1147–1161, apr 2017.

[29] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1082–1090, 2011.

[30] A. K. Jain, M. N. Murty, and P. J. Flynn, "Estimating origin-destination flows using mobile phone location data," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[31] H. Zang and J. C. Bolot, "Mining call and mobility data to improve paging efficiency in cellular networks," in *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pp. 123–134, 2007.

[32] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility.," *Science (New York, N.Y.)*, vol. 327, pp. 1018–21, feb 2010.

[33] Y. Li, T. Wu, P. Hui, D. Jin, and S. Chen, "Social-aware D2D communications: Qualitative insights and quantitative analysis," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 150–158, 2014.

[34] M. Chen, Y. Hao, M. Qiu, J. Song, D. Wu, and I. Humar, "Mobility-Aware Caching and Computation Offloading in 5G Ultra-Dense Cellular Networks," *Sensors*, vol. 16, p. 974, jun 2016.

[35] M. Chen, Y. Hao, L. Hu, K. Huang, and V. K. N. Lau, "Green and Mobility-Aware Caching in 5G Networks," *IEEE Transactions on Wireless Communications*, vol. 16, pp. 8347–8361, dec 2017.

[36] D. Ren, X. Gui, K. Zhang, and J. Wu, "Mobility-Aware Traffic Offloading via Cooperative Coded Edge Caching," *IEEE Access*, vol. 8, pp. 43427–43442, 2020.

[37] J. Hu, W. Heng, G. Zhang, and C. Meng, "Base station sleeping mechanism based on traffic prediction in heterogeneous networks," in *2015 International Telecommunication Networks and Applications Conference (ITNAC)*, pp. 83–87, IEEE, 2015.

[38] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of Research and Development*, vol. 3, pp. 210–229, jul 1959.

[39] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[40] M. Minsky and S. Papert, "Perceptrons.," 1969.

[41] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[42] B. Marr, "A short history of machine learning–every manager should read," *Forbes. http://tinyurl. com/gslvr6k*, 2016.

[43] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[44] H. T. Siegelmann and E. D. Sontag, "On the computational power of neural nets," *Journal of computer and system sciences*, vol. 50, no. 1, pp. 132–150, 1995.

[45] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[46] S. Russell and P. Norvig, "Artificial intelligence: a modern approach," 2002.

[47] C. Zhang, P. Patras, and H. Haddadi, "Deep Learning in Mobile and Wireless Networking: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2224–2287, 2019.

[48] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, "Application of machine learning in wireless networks: Key techniques and open issues," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3072–3108, 2019.

[49] N. Wang, E. Hossain, and V. K. Bhargava, "Backhauling 5G small cells: A radio resource management perspective," *IEEE Wireless Communications*, vol. 22, no. 5, pp. 41–49, 2015.

[50] F. Giust, L. Cominardi, and C. J. Bernardos, "Distributed mobility management for future 5G networks: overview and analysis of existing approaches," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 142–149, 2015.

[51] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.

[52] C. Zhang, H. Zhang, J. Qiao, D. Yuan, and M. Zhang, "Deep Transfer Learning for Intelligent Cellular Traffic Prediction Based on Cross-Domain Big Data," *IEEE Journal on Selected Areas in Communications*, vol. 37, pp. 1389–1401, jun 2019.

[53] D. Fagen, P. A. Vicharelli, and J. Weitzen, "Automated Wireless Coverage Optimization With Controlled Overlap," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 4, pp. 2395–2403, 2008.

[54] A. Lobinger, S. Stefanski, T. Jansen, and I. Balan, "Coordinating Handover Parameter Optimization and Load Balancing in LTE Self-Optimizing Networks," in *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*, pp. 1–5, 2011.

[55] A. Aguilar-Garcia, S. Fortes, A. F. Duran, and R. Barco, "Context-Aware Self-Optimization: Evolution Based on the Use Case of Load Balancing in Small-Cell Networks," *IEEE Vehicular Technology Magazine*, vol. 11, no. 1, pp. 86–95, 2016.

[56] B. Ma, W. Guo, and J. Zhang, "A Survey of Online Data-Driven Proactive 5G Network Optimisation Using Machine Learning," *IEEE Access*, vol. 8, pp. 35606–35637, 2020.

[57] A. Patil and H. K. Sawant, "Technical specification group services and system aspects, IP multimedia subsystem (IMS)," *Int. J. Electron. Commun. Comput. Eng.*, vol. 3, no. 2, pp. 234–238, 2012.

[58] M. A. Alsheikh, D. Niyato, S. Lin, H.-P. Tan, and Z. Han, "Mobile big data analytics using deep learning and apache spark," *IEEE network*, vol. 30, no. 3, pp. 22–29, 2016.

[59] H. Zhu, Y. Zhang, M. Li, A. Ashok, and K. Ota, "Exploring deep learning for efficient and reliable mobile sensing," *IEEE Network*, vol. 32, no. 4, pp. 6–7, 2018.

[60] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, "Machine learning for networking: Workflow, advances and opportunities," *IEEE Network*, vol. 32, no. 2, pp. 92–99, 2017.

[61] R. Li, Z. Zhao, X. Zhou, G. Ding, Y. Chen, Z. Wang, and H. Zhang, "Intelligent 5G: When Cellular Networks Meet Artificial Intelligence," *IEEE Wireless Communications*, vol. 24, pp. 175–183, oct 2017.

[62] C. Lynch, "How do your data grow?," *Nature*, vol. 455, no. 7209, pp. 28–29, 2008.

[63] T. J. Barnett, A. Sumits, S. Jain, and U. Andra, "Cisco Visual Networking Index (VNI) Update Global Mobile Data Traffic Forecast," *Vni*, pp. 2015–2020, 2015.

[64] J. Clement, "Global mobile data traffic 2017-2022," *Statista, Available: https://www. statista. com/statistics/271405/global-mobile-datatraffic-forecast/(Accessed December 2020)*, 2019.

[65] Qualcomm, "The 1000x data challenge," 2013.

[66] P. K. Agyapong, M. Iwamura, D. Staehle, W. Kiess, and A. Benjebbour, "Design considerations for a 5G network architecture," *IEEE Communications Magazine*, vol. 52, no. 11, pp. 65–75, 2014.

[67] "3GPP work items on self-organizing networks, v0.1.3 (2014-06)," 2014.

[68] I. Allal, B. Mongazon-Cazavet, K. A. Agha, S. Senouci, and Y. Gourhant, "A green small cells deployment in 5G — Switch ON/OFF via IoT networks & energy efficient mesh backhauling," in *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, pp. 1–2, 2017.

[69] B. Chandrasekaran, "Survey of network traffic models," *Waschington University in St. Louis CSE*, vol. 567, 2009.

[70] P. Semov, P. Koleva, and V. Poulkov, "Adaptive resource scheduling based on neural network and mobile traffic prediction," in *2019 42nd International Conference on Telecommunications and Signal Processing, TSP 2019*, pp. 585–588, Institute of Electrical and Electronics Engineers Inc., jul 2019.

[71] I. Angri, M. Mahfoudi, A. Najid, and M. El Bekkali, "Exponential MLWDF (EXP-MLWDF) downlink scheduling algorithm evaluated in LTE for high mobility and dense area scenario," *International Journal of Electrical and Computer Engineering*, vol. 8, no. 3, p. 1618, 2018.

[72] V. Sciancalepore, K. Samdanis, X. Costa-Perez, D. Bega, M. Gramaglia, and A. Banchs, "Mobile traffic forecasting for maximizing 5G network slicing resource utilization," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pp. 1–9, 2017.

[73] N. Saxena, B. J. Sahu, and Y. S. Han, "Traffic-aware energy optimization in green LTE cellular systems," *IEEE Communications Letters*, vol. 18, pp. 38–41, jan 2014.

[74] H. D. Trinh, L. Giupponi, and P. Dini, "Mobile Traffic Prediction from Raw Data Using LSTM Networks," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1827–1832, 2018.

[75] C. Williamson, E. Halepovic, H. Sun, and Y. Wu, "Characterization of CDMA2000 cellular data network traffic," in *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05) l*, pp. Z000–719, IEEE, 2005.

[76] M. Z. Shafiq, L. Ji, A. X. Liu, and J. Wang, "Characterizing and modeling internet traffic dynamics of cellular devices," *ACM SIGMETRICS Performance Evaluation Review*, vol. 39, no. 1, pp. 265–276, 2011.

[77] G. Xiaohu, S. Yu, W.-S. Yoon, and Y.-D. Kim, "A new prediction method of alpha-stable processes for self-similar traffic," in *IEEE Global Telecommunications Conference, 2004. GLOBECOM '04.*, vol. 2, pp. 675–679 Vol.2, 2004.

[78] D. Tikunov and T. Nishimura, "Traffic prediction for mobile network using Holt-Winter's exponential smoothing," in *2007 15th International Conference on Software, Telecommunications and Computer Networks*, pp. 1–5, 2007.

[79] F. Xu, Y. Lin, J. Huang, D. Wu, H. Shi, J. Song, and Y. Li, "Big Data Driven Mobile Traffic Understanding and Forecasting: A Time Series Approach," *IEEE Transactions on Services Computing*, vol. 9, pp. 796–805, sep 2016.

[80] H. Sun, H. X. Liu, H. Xiao, and B. Ran, "Short term traffic forecasting using the local linear regression model," 2002.

[81] R. Li, Z. Zhao, X. Zhou, and H. Zhang, "Energy savings scheme in radio access networks via compressive sensing-based traffic load prediction," *Transactions on Emerging Telecommunications Technologies*, vol. 25, pp. 468–478, apr 2014.

[82] R. Li, Z. Zhao, Y. Wei, X. Zhou, and H. Zhang, "GM-PAB: A grid-based energy saving scheme with predicted traffic load guidance for cellular networks," in *IEEE International Conference on Communications*, pp. 1160–1164, 2012.

[83] J. Biesterfeld, E. Ennigrou, and K. Jobmann, "Neural networks for location prediction in mobile networks," in *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications (IWANNT'97)*, pp. 207–214, 1997.

[84] Y. Tian and L. Pan, "Predicting Short-Term Traffic Flow by Long Short-Term Memory Recurrent Neural Network," in *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, pp. 153–158, 2015.

[85] Yantai Shu, Minfang Yu, Jiakun Liu, and O. Yang, "Wireless traffic modeling and prediction using seasonal ARIMA models," in *IEEE International Conference on Communications, 2003. ICC '03.*, vol. 3, pp. 1675–1679, IEEE.

[86] J. Guo, Y. Peng, X. Peng, Q. Chen, J. Yu, and Y. Dai, "Traffic forecasting for mobile networks with multiplicative seasonal ARIMA models," in *2009 9th International Conference on Electronic Measurement & Instruments*, pp. 3–380, 2009.

[87] R. Li, Z. Zhao, X. Zhou, J. Palicot, and H. Zhang, "The prediction analysis of cellular radio access network traffic: From entropy theory to networking practice," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 234–240, 2014.

[88] F. Ju, J. Yang, and H. Liu, "Analysis of Self-Similar Traffic Based on the On/Off Model," in *2009 International Workshop on Chaos-Fractals Theories and Applications*, pp. 301–304, 2009.

[89] S. Wang, X. Zhang, J. Zhang, J. Feng, W. Wang, and K. Xin, "An Approach for Spatial-Temporal Traffic Modeling in Mobile Cellular Networks," in *2015 27th International Teletraffic Congress*, pp. 203–209, 2015.

[90] techopedia, "Network Traffic," *https://www.techopedia.com/definition/29917/network-traffic*, 2015.

[91] X. Wang, A. V. Vasilakos, M. Chen, Y. Liu, and T. T. Kwon, "A Survey of Green Mobile Networks: Opportunities and Challenges," *Mobile Networks and Applications*, vol. 17, pp. 4–20, feb 2012.

[92] C. V. N. Index, "Forecast and methodology, 2016–2021," *White Paper, June*, 2017.

[93] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. YANG, and W. Wang, "A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[94] G. Soos, D. Ficzere, and P. Varga, "Towards Traffic Identification and Modeling for 5G Application Use-Cases," *Electronics*, vol. 9, no. 4, p. 640, 2020.

[95] R. Keralapura, A. Nucci, Z.-L. Zhang, and L. Gao, "Profiling users in a 3g network using hourglass co-clustering," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pp. 341–352, 2010.

[96] Y. Zhang and A. Årvidsson, "Understanding the characteristics of cellular data traffic," in *Proceedings of the 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design*, pp. 13–18, 2012.

[97] E. M. R. Oliveira, A. C. Viana, K. P. Naveen, and C. Sarraute, "Measurement-driven mobile data traffic modeling in a large metropolitan area," in *2015 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 230–235, IEEE, 2015.

[98] Y. Wang, M. Faloutsos, and H. Zang, "On the usage patterns of multimodal communication: Countries and evolution," in *2013 Proceedings IEEE INFOCOM*, pp. 3135–3140, IEEE, 2013.

[99] S. Hoteit, S. Secci, Z. He, C. Ziemlicki, Z. Smoreda, C. Ratti, and G. Pujolle, "Content consumption cartography of the paris urban region using cellular probe data," in *Proceedings of the first workshop on Urban networking*, pp. 43–48, 2012.

[100] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, and J. Wang, "Characterizing geospatial dynamics of application usage in a 3G cellular data network," in *2012 Proceedings IEEE INFOCOM*, pp. 1341–1349, IEEE, 2012.

[101] Y. Jin, N. Duffield, A. Gerber, P. Haffner, W.-L. Hsu, G. Jacobson, S. Sen, S. Venkataraman, and Z.-L. Zhang, "Characterizing data usage patterns in a large cellular network," in *Proceedings of the 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design*, pp. 7–12, 2012.

[102] B. Zhou, D. He, and Z. Sun, "Traffic modeling and prediction using ARIMA/GARCH model," in *Modeling and Simulation Tools for Emerging Telecommunication Networks: Needs, Trends, Challenges and Solutions*, pp. 101–121, Springer Science and Business Media, LLC, 2006.

[103] Y. Shu, M. Yu, J. Liu, O. W. W. Yang, Yantai Shu, Minfang Yu, Jiakun Liu, and O. W. W. Yang, "Wireless traffic modeling and prediction using seasonal ARIMA models," in *IEEE International Conference on Communications, 2003. ICC '03.*, vol. 3, pp. 1675–1679 vol.3, IEEE, 2003.

[104] A. Hess, K. A. Hummel, W. N. Gansterer, and G. Haring, "Data-driven human mobility modeling: a survey and engineering guidance for mobile networking," *ACM Computing Surveys (CSUR)*, vol. 48, no. 3, pp. 1–39, 2015.

[105] A. Janecek, K. A. Hummel, D. Valerio, F. Ricciato, and H. Hlavacs, "Cellular data meet vehicular traffic theory: location area updates and cell transitions for travel time estimation," in *Proceedings of the 2012 ACM conference on ubiquitous computing*, pp. 361–370, 2012.

[106] A. Nadembega, A. Hafid, and T. Taleb, "Mobility-Prediction-Aware Bandwidth Reservation Scheme for Mobile Networks," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 6, pp. 2561–2576, 2015.

[107] P. V. Klaine, M. A. Imran, O. Onireti, and R. D. Souza, "A Survey of Machine Learning Techniques Applied to Self Organizing Cellular Networks," *IEEE Communications Surveys & Tutorials*, pp. 1–1, 2017.

[108] H. Zhang and L. Dai, "Mobility Prediction: A Survey on State-of-the-Art Schemes and Future Applications," *IEEE Access*, vol. 7, pp. 802–822, 2019.

[109] D. Tang and M. Baker, "Analysis of a metropolitan-area wireless network," *Wireless Networks*, vol. 8, no. 2-3, pp. 107–120, 2002.

[110] J.-P. Onnela, S. Arbesman, M. C. González, A.-L. Barabási, and N. A. Christakis, "Geographic constraints on social network groups," *PLoS one*, vol. 6, no. 4, p. e16939, 2011.

[111] R. Lambiotte, V. D. Blondel, C. De Kerchove, E. Huens, C. Prieur, Z. Smoreda, and P. Van Dooren, "Geographical dispersal of mobile communication networks," *Physica A: Statistical Mechanics and its Applications*, vol. 387, no. 21, pp. 5317–5325, 2008.

[112] D. Wang and C. Song, "Impact of human mobility on social networks," *Journal of Communications and Networks*, vol. 17, no. 2, pp. 100–109, 2015.

[113] Y. C. Chen, L. Qiu, Y. Zhang, G. Xue, and Z. Hu, "Robust network compressive sensing," in *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, (New York, New York, USA), pp. 545–556, Association for Computing Machinery, sep 2014.

[114] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu, "Spatio-temporal compressive sensing and internet traffic matrices," in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication - SIGCOMM '09*, (New York, New York, USA), p. 267, Association for Computing Machinery (ACM), 2009.

[115] N. Sapankevych and R. Sankar, "Time series prediction using support vector machines: A survey," *IEEE Computational Intelligence Magazine*, vol. 4, pp. 24–38, may 2009.

[116] R. H. Filho and J. E. B. Maia, "Network traffic prediction using PCA and K-means," in *2010 IEEE Network Operations and Management Symposium - NOMS 2010*, pp. 938–941, 2010.

[117] A. Soule, A. Lakhina, N. Taft, K. Papagiannaki, K. Salamatian, A. Nucci, M. Crovella, and C. Diot, "Traffic matrices: Balancing measurements, inference and modeling," in *Performance Evaluation Review*, vol. 33, (New York, New York, USA), pp. 362–373, ACM Press, 2005.

[118] M. C. Falvo, M. Gastaldi, A. Nardecchia, and A. Prudenzi, "Kalman filter for short-term load forecasting: An hourly predictor of municipal load," 2007.

[119] Y. Xu, F. Yin, W. Xu, J. Lin, and S. Cui, "Wireless Traffic Prediction With Scalable Gaussian Process: Framework, Algorithms, and Verification," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1291–1306, 2019.

[120] L. Nie, D. Jiang, S. Yu, and H. Song, "Network traffic prediction based on deep belief network in wireless mesh backbone networks," in *IEEE Wireless Communications and Networking Conference, WCNC*, Institute of Electrical and Electronics Engineers Inc., may 2017.

[121] X. Wang, Z. Zhou, F. Xiao, K. Xing, Z. Yang, Y. Liu, and C. Peng, "Spatio-Temporal Analysis and Prediction of Cellular Traffic in Metropolis," *IEEE Transactions on Mobile Computing*, vol. 18, no. 9, pp. 2190–2202, 2019.

[122] C. Qiu, Y. Zhang, Z. Feng, P. Zhang, and S. Cui, "Spatio-Temporal Wireless Traffic Prediction with Recurrent Neural Network," *IEEE Wireless Communications Letters*, vol. 7, pp. 554–557, aug 2018.

[123] Y. Hua, Z. Zhao, Z. Liu, X. Chen, R. Li, and H. Zhang, "Traffic Prediction Based on Random Connectivity in Deep Learning with Long Short-Term Memory," in *IEEE Vehicular Technology Conference*, vol. 2018-August, Institute of Electrical and Electronics Engineers Inc., jul 2018.

[124] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang, "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach," pp. 1–9, 2017.

[125] C. Zhang, X. Ouyang, and P. Patras, "ZipNet-GAN: Inferring fine-grained mobile traffic patterns via a generative adversarial neural network," in *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*, pp. 363–375, 2017.

[126] C.-W. Huang, C.-T. Chiang, and Q. Li, "A study of deep learning networks on mobile traffic forecasting," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–6, IEEE, 2017.

[127] S. Akoush and A. Sameh, "Mobile user movement prediction using bayesian learning for neural networks," in *Proceedings of the 2007 international conference on Wireless communications and mobile computing*, pp. 191–196, 2007.

[128] M. Chen, M. Mozaffari, W. Saad, C. Yin, M. Debbah, and C. S. Hong, "Caching in the Sky: Proactive Deployment of Cache-Enabled Unmanned Aerial Vehicles for Optimized Quality-of-Experience," *IEEE Journal on Selected Areas in Communications*, vol. 35, pp. 1046–1061, may 2017.

[129] J. Yang, C. Dai, and Z. Ding, "A scheme of terminal mobility prediction of Ultra Dense Network based on SVM," in *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)(*, pp. 837–842, IEEE, mar 2017.

[130] N. Eagle and A. S. Pentland, "Eigenbehaviors: identifying structure in routine," *Behavioral Ecology and Sociobiology*, vol. 63, pp. 1057–1066, may 2009.

[131] J. Reades, F. Calabrese, and C. Ratti, "Eigenplaces: analysing cities using the space–time structure of the mobile phone network," *Environment and Planning B: Planning and Design*, vol. 36, no. 5, pp. 824–836, 2009.

[132] F. Calabrese, J. Reades, and C. Ratti, "Eigenplaces: segmenting space through digital signatures," *IEEE Pervasive Computing*, vol. 9, no. 1, pp. 78–84, 2009.

[133] X. Ouyang, C. Zhang, P. Zhou, H. Jiang, and S. Gong, "Deepspace: An online deep learning framework for mobile big data to understand human mobility patterns," *arXiv preprint arXiv:1610.07009*, 2016.

[134] Nam Tuan Nguyen, Yichuan Wang, Husheng Li, Xin Liu, and Zhu Han, "Extracting typical users' moving patterns using deep learning," in *2012 IEEE Global Communications Conference (GLOBECOM)*, pp. 5410–5414, IEEE, dec 2012.

[135] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Artificial Neural Networks-Based Machine Learning for Wireless Networks: A Tutorial," oct 2017.

[136] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[137] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.

[138] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[139] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," 2014.

[140] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A Novel Connectionist System for Unconstrained Handwriting Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–868, 2009.

[141] A. Singh, "Anomaly detection for temporal data using long short-term memory (lstm)," 2017.

[142] W. Duch and N. Jankowski, "Survey of neural transfer functions," *Neural Computing Surveys*, vol. 2, no. 1, pp. 163–212, 1999.

[143] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[144] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: driving directions based on taxi trajectories," in *Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems*, pp. 99–108, 2010.

[145] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 316–324, 2011.

[146] Y. Zheng, Y. Chen, X. Xie, and W.-Y. Ma, "GeoLife2. 0: a location-based social networking service," in *2009 tenth international conference on mobile data management: systems, services and middleware*, pp. 357–358, IEEE, 2009.

[147] P. Simon, *Too big to ignore : the business case for big data*.

[148] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[149] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining correlation between locations using human location history," in *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pp. 472–475, 2009.

[150] "• Number of social media users worldwide 2010-2021 | Statista."

[151] T. Qiu, J. Feng, Z. Ge, J. Wang, J. Xu, and J. Yates, "Listen to me if you can: tracking user experience of mobile network on social media," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pp. 288–293, ACM, 2010.

[152] K. Takeshita, M. Yokota, and K. Nishimatsu, "Early network failure detection system by analyzing Twitter data," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 279–286, IEEE, 2015.

[153] W. Guo and J. Zhang, "Uncovering wireless blackspots using Twitter data," *Electronics Letters*, vol. 53, pp. 814–816, jun 2017.

[154] N. Kumar, "Sentiment Analysis of Twitter Messages: Demonetization a Use Case," in *2nd International Conference on Computational Systems and Information Technology for Sustainable Solutions, CSITSS 2017*, Institute of Electrical and Electronics Engineers Inc., aug 2018.

[155] E. Miranda, M. Aryuni, R. Hariyanto, and E. S. Surya, "Sentiment Analysis using Sentiwordnet and Machine Learning Approach (Indonesia general election opinion from the twitter content)," in *Proceedings of 2019 International Conference on Information Management and Technology, ICIMTech 2019*, pp. 62–67, Institute of Electrical and Electronics Engineers Inc., aug 2019.

[156] R. Jose and V. S. Chooralil, "Prediction of election result by enhanced sentiment analysis on Twitter data using Word Sense Disambiguation," in *2015 International Conference on Control, Communication and Computing India, ICCC 2015*, pp. 638–641, Institute of Electrical and Electronics Engineers Inc., mar 2016.

[157] A. S. Pereira, T. R. M. B. Silva, F. A. Silva, and A. A. F. Loureiro, "Traffic Event Detection Using Online Social Networks," in *2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pp. 61–64, IEEE, jun 2017.

[158] A. Y. Nikravesh, S. A. Ajila, C. H. Lung, and W. Ding, "Mobile network traffic prediction using MLP, MLPWD, and SVM," in *Proceedings - 2016 IEEE International Congress on Big Data, BigData Congress 2016*, pp. 402–409, Institute of Electrical and Electronics Engineers Inc., oct 2016.

[159] A. Azzouni and G. Pujolle, "A Long Short-Term Memory Recurrent Neural Network Framework for Network Traffic Matrix Prediction," may 2017.

[160] "• Twitter: number of active users 2010-2019 | Statista."

[161] B. Yang, W. Guo, B. Chen, G. Yang, and J. Zhang, "Estimating Mobile Traffic Demand Using Twitter," *IEEE Wireless Communications Letters*, vol. 5, pp. 380–383, aug 2016.

[162] R. Li, Z. Zhao, J. Zheng, C. Mei, Y. Cai, and H. Zhang, "The Learning and Prediction of Application-Level Traffic Data in Cellular Networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3899–3912, 2017.

[163] G. Barlacchi, M. De Nadai, R. Larcher, A. Casella, C. Chitic, G. Torrisi, F. Antonelli, A. Vespignani, A. Pentland, and B. Lepri, "A multi-source dataset of urban life in the city of Milan and the Province of Trentino," *Scientific data*, vol. 2, p. 150055, oct 2015.

[164] C. Zhang, H. Zhang, D. Yuan, and M. Zhang, "Citywide Cellular Traffic Prediction Based on Densely Connected Convolutional Neural Networks," *IEEE Communications Letters*, vol. 22, pp. 1656–1659, aug 2018.

[165] J. Schmidhuber, "Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook," 1987.

[166] C. Lemke, M. Budka, and B. Gabrys, "Metalearning: a survey of trends and technologies," *Artificial intelligence review*, vol. 44, no. 1, pp. 117–130, 2015.

[167] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, and J. Liu, "LSTM network: a deep learning approach for short-term traffic forecast," *IET Intelligent Transport Systems*, vol. 11, no. 2, pp. 68–75, 2017.

[168] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[169] Y. Huang, J. Tan, and Y.-C. Liang, "Wireless big data: transforming heterogeneous networks to smart networks," *Journal of Communications and Information Networks*, vol. 2, no. 1, pp. 19–32, 2017.

[170] C. V. N. Index, "Global mobile data traffic forecast update, 2016–2021," *white paper*, 2017.

[171] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Toward Effective Mobile Encrypted Traffic Classification through Deep Learning," *Neurocomputing*, 2020.

[172] J. Feng, X. Chen, R. Gao, M. Zeng, and Y. Li, "Deeptp: An end-to-end neural network for mobile cellular traffic prediction," *IEEE Network*, vol. 32, no. 6, pp. 108–115, 2018.

Appendix A

Proof of Lemma 1

For the two discrete periodic signals $l_1[t]$ and $l_2[t]$ with the period of T, we can represent their frequency components by applying FFT as:

$$F_1(k \cdot \frac{2\pi}{T}) = FFT[l_1[t]] = \frac{1}{T}\sum_{t=0}^{T-1} l_1[t] W_T^{kt} \tag{A.1}$$

$$F_2(k \cdot \frac{2\pi}{T}) = FFT[l_2[t]] = \frac{1}{T}\sum_{t=0}^{T-1} l_2[t] W_T^{kt} \tag{A.2}$$

where $W_T = e^{-j\frac{2\pi}{T}}$

We can also represent $l_1[t]$ and $l_2[t]$ for $t \in Z$ by their Inverse Fast Fourier Transform (IFFT):

$$l_1[t] = \sum_{k=0}^{T-1} F_1(k \cdot \frac{2\pi}{T}) \cdot W_T^{-kt} \tag{A.3}$$

$$l_2[t] = \sum_{k=0}^{T-1} F_2(k \cdot \frac{2\pi}{T}) \cdot W_T^{-kt} \tag{A.4}$$

Obviously, for an arbitrary $t \in Z$, we have:

$$
\begin{aligned}
&|l_1[t]-l_2[t]| = \left|\sum_{k=0}^{T-1} F_1(k\cdot\tfrac{2\pi}{T})\cdot W_T^{-kt} - \sum_{k=0}^{T-1} F_2(k\cdot\tfrac{2\pi}{T})\cdot W_T^{-kt}\right| \\
&= \left|\sum_{k\cdot\frac{2\pi}{T}\in f_{main}} [F_1(k\cdot\tfrac{2\pi}{T}) - F_2(k\cdot\tfrac{2\pi}{T})]\cdot W_T^{-kt} + \sum_{k\cdot\frac{2\pi}{T}\notin f_{main},\, k\cdot\frac{2\pi}{T}\in[0,2\pi)} [F_1(k\cdot\tfrac{2\pi}{T}) - F_2(k\cdot\tfrac{2\pi}{T})]\cdot W_T^{-kt}\right| \\
&\le \left|\sum_{k\cdot\frac{2\pi}{T}\in f_{main}} [F_1(k\cdot\tfrac{2\pi}{T}) - F_2(k\cdot\tfrac{2\pi}{T})]\cdot W_T^{-kt}\right| + \left|\sum_{k\cdot\frac{2\pi}{T}\notin f_{main},\, k\cdot\frac{2\pi}{T}\in[0,2\pi)} [F_1(k\cdot\tfrac{2\pi}{T}) - F_2(k\cdot\tfrac{2\pi}{T})]\cdot W_T^{-kt}\right|
\end{aligned}
\tag{A.5}
$$

Since $\left|W_T^{-kt}\right|^2 = 1$ for arbitrary $k = 1, ..., T-1$, we have the following inequalities based on the inequality of arithmetic and geometric means:

$$
\begin{aligned}
&\left|\sum_{k\cdot\frac{2\pi}{T}\in f_{main}} [F_1(k\cdot\tfrac{2\pi}{T}) - F_2(k\cdot\tfrac{2\pi}{T})]\cdot W_T^{-kt}\right| \le \sum_{k\cdot\frac{2\pi}{T}\in f_{main}} \left|[F_1(k\cdot\tfrac{2\pi}{T}) - F_2(k\cdot\tfrac{2\pi}{T})]\cdot W_T^{-kt}\right| \\
&\le \sqrt{k}\cdot\sqrt{\sum_{k\cdot\frac{2\pi}{T}\in f_{main}} \left|[F_1(k\cdot\tfrac{2\pi}{T}) - F_2(k\cdot\tfrac{2\pi}{T})]\cdot W_T^{-kt}\right|^2} = \sqrt{k}\cdot\sqrt{\sum_{k\cdot\frac{2\pi}{T}\in f_{main}} \left|[F_1(k\cdot\tfrac{2\pi}{T}) - F_2(k\cdot\tfrac{2\pi}{T})]\right|^2} \\
&= \sqrt{k}\cdot\sqrt{\sum_{k\cdot\frac{2\pi}{T}\in f_{main}} [\Re(F_1(k\cdot\tfrac{2\pi}{T})) - \Re(F_2(k\cdot\tfrac{2\pi}{T}))]^2 + \sum_{k\cdot\frac{2\pi}{T}\in f_{main}} [\Im(F_1(k\cdot\tfrac{2\pi}{T})) - \Im(F_2(k\cdot\tfrac{2\pi}{T}))]^2} \\
&= \sqrt{k}\cdot\sigma
\end{aligned}
\tag{A.6}
$$

For $\left|\sum_{k\cdot\frac{2\pi}{T}\notin f_{main},\, k\cdot\frac{2\pi}{T}\in[0,2\pi)} [F_1(k\cdot\tfrac{2\pi}{T}) - F_2(k\cdot\tfrac{2\pi}{T})]\cdot W_T^{-kt}\right|$, we have:

$$
\begin{aligned}
&\left|\sum_{k\cdot\frac{2\pi}{T}\notin f_{main},\, k\cdot\frac{2\pi}{T}\in[0,2\pi)} [F_1(k\cdot\tfrac{2\pi}{T}) - F_2(k\cdot\tfrac{2\pi}{T})]\cdot W_T^{-kt}\right| \\
&\le \left|\sum_{k\cdot\frac{2\pi}{T}\notin f_{main},\, k\cdot\frac{2\pi}{T}\in[0,2\pi)} F_1(k\cdot\tfrac{2\pi}{T})\cdot W_T^{-kt}\right| + \left|\sum_{k\cdot\frac{2\pi}{T}\notin f_{main},\, k\cdot\frac{2\pi}{T}\in[0,2\pi)} F_2(k\cdot\tfrac{2\pi}{T})\cdot W_T^{-kt}\right| \\
&\le \sum_{k\cdot\frac{2\pi}{T}\notin f_{main},\, k\cdot\frac{2\pi}{T}\in[0,2\pi)} \left|F_1(k\cdot\tfrac{2\pi}{T})\cdot W_T^{-kt}\right| + \sum_{k\cdot\frac{2\pi}{T}\notin f_{main},\, k\cdot\frac{2\pi}{T}\in[0,2\pi)} \left|F_2(k\cdot\tfrac{2\pi}{T})\cdot W_T^{-kt}\right| \\
&= \sum_{k\cdot\frac{2\pi}{T}\notin f_{main},\, k\cdot\frac{2\pi}{T}\in[0,2\pi)} \left|F_1(k\cdot\tfrac{2\pi}{T})\right| + \sum_{k\cdot\frac{2\pi}{T}\notin f_{main},\, k\cdot\frac{2\pi}{T}\in[0,2\pi)} \left|F_2(k\cdot\tfrac{2\pi}{T})\right| \\
&\le 2\cdot\eta
\end{aligned}
\tag{A.7}
$$

Combining (A.5) with (A.6) and (A.7), we have:

$$|l_1[t] - l_2[t]| \leq \sqrt{k} \cdot \sigma + 2 \cdot \eta, \ \forall t \in Z \tag{A.8}$$

We arrive at **Lemma 1**.

Appendix B

Proof of Lemma 2

For an LSTM block shown in Fig. 12, we denote the input vectors of the forget gate, input gate, input activation gate and output gate as $\mathbf{f}^i[t] = (f_1^i[t], ..., f_v^i[t])$, $\mathbf{i}^i[t] = (i_1^i[t], ..., i_v^i[t])$, $\mathbf{z}^i[t] = (z_1^i[t], ..., z_v^i[t])$, and $\mathbf{o}^i[t] = (o_1^i[t], ..., o_v^i[t])$, respectively. Correspondingly, we use $\mathbf{f}^o[t] = (f_1^o[t], ..., f_v^o[t])$, $\mathbf{i}^o[t] = (i_1^o[t], ..., i_v^o[t])$, $\mathbf{z}^o[t] = (z_1^o[t], ..., z_v^o[t])$, and $\mathbf{o}^o[t] = (o_1^o[t], ..., o_v^o[t])$, respectively, to denote the output vectors of the logical gates. We use $\mathbf{Z}[t] = (Z_1[t], ..., Z_v[t])$ to denote the status vector of the LSTM block. We use the $(U+V) \times V$ matrixes $\mathbf{W}_f$, $\mathbf{W}_i$, $\mathbf{W}_z$, and $\mathbf{W}_o$, respectively to denote the neural connection weights in the forget gate, input gate, input activation gate and output gate, while use the V-dimensional vectors $\mathbf{b}_f$, $\mathbf{b}_i$, $\mathbf{b}_z$, and $\mathbf{b}_o$, respectively, to denote the neural biases in the forget gate, input gate, input activation gate and output gate.

For an arbitrary element b in the output vector, we have the following equations according to the structure of the LSTM block:

$$
\begin{aligned}
&h_v[t-SN+1] = \varphi(i_v^o[t-SN+1] \cdot z_v^o[t-SN+1]) \cdot o_v^o[t-SN+1] \\
&= \varphi(\sigma(i_v^i[t-SN+1] + \mathbf{b}_i(v)) \cdot \varphi(z_v^i[t-SN+1] + \mathbf{b}_c(v))) \cdot \sigma(o_v^i[t-SN+1] + \mathbf{b}_o(v)) \\
&= \varphi(\sigma(\sum_{u^*=1}^{U} x_{u^*}[t-SN+1] \cdot \mathbf{W}_i(V+u^*, v) + \mathbf{b}_i(v)) \cdot \varphi(\sum_{u^*=1}^{U} x_{u^*}[t-SN+1] \cdot \mathbf{W}_c(V+u^*, v) + \mathbf{b}_c(v))) \\
&\cdot \sigma(\sum_{u^*=1}^{U} x_{u^*}[t-SN+1] \cdot \mathbf{W}_o(V+u^*, v) + \mathbf{b}_o(v))
\end{aligned}
\tag{B.1}
$$

$$
\begin{aligned}
&Z_v[t-SN+1] = i_v^o[t-SN+1] \cdot z_v^o[t-SN+1] \\
&= \sigma(\sum_{u^*=1}^{U} x_{u^*}[t-SN+1] \cdot \mathbf{W}_i(V+u^*,v) + \mathbf{b}_i(v)) \cdot \varphi(\sum_{u^*=1}^{U} x_{u^*}[t-SN+1] \cdot \mathbf{W}_c(V+u^*,v) + \mathbf{b}_c(v))
\end{aligned} \tag{B.2}
$$

Obviously, $h_v[t-SN+1]$ and $Z_v[t-SN+1]$ are continuous functions about $x_1[t-SN+1]$, ..., $x_U[t-SN+1]$. For an arbitrary element a in the LSTM block's input vector, the partial derivatives of $h_v[t-SN+1]$ and $z_v[t-SN+1]$ w.r.t. $x_u[t-SN+1]$ can be calculated as:

$$
\begin{aligned}
&\frac{\partial Z_v[t-SN+1]}{\partial x_u[t-SN+1]} = \frac{\partial \sigma(i_v^i[t-SN+1]+\mathbf{b}_i(v))}{\partial x_u[t-SN+1]} \cdot z_v^o[t-SN+1] + i_v^o[t-SN+1] \cdot \frac{\partial \varphi(z_v^i[t-SN+1]+\mathbf{b}_c(v))}{\partial x_u[t-SN+1]} \\
&= \sigma(i_v^i[t-SN+1]+\mathbf{b}_i(v)) \cdot (1-\sigma(i_v^i[t-SN+1]+\mathbf{b}_i(v)) \cdot \mathbf{W}_i(V+a,v) \cdot z_v^o[t-SN+1] \\
&+ i_v^o[t-SN+1] \cdot (1-\varphi^2(z_v^i[t-SN+1]+\mathbf{b}_c(v))) \cdot \mathbf{W}_c(V+a,v)
\end{aligned} \tag{B.3}
$$

$$
\begin{aligned}
&\frac{\partial h_v[t-SN+1]}{\partial x_u[t-SN+1]} = \frac{\partial \varphi(i_v^o[t-SN+1] \cdot z_v^o[t-SN+1])}{\partial x_u[t-SN+1]} \cdot o_v^o[t-SN+1] + \varphi(i_v^o[t-SN+1] \cdot z_v^o[t-SN+1]) \cdot \frac{\partial o_v^o[t-SN+1]}{\partial x_u[t-SN+1]} \\
&= (1-\varphi^2(i_v^o[t-SN+1] \cdot z_v^o[t-SN+1])) \cdot o_v^o[t-SN+1] \\
&\cdot (\frac{\partial \sigma(i_v^i[t-SN+1]+\mathbf{b}_i(v))}{\partial x_u[t-SN+1]} \cdot z_v^o[t-SN+1] + i_v^o[t-SN+1] \cdot \frac{\partial \varphi(z_v^i[t-SN+1]+\mathbf{b}_c(v))}{\partial x_u[t-SN+1]}) \\
&+ \varphi(i_v^o[t-SN+1] \cdot z_v^o[t-SN+1]) \cdot \frac{\partial \sigma(o_v^i[t-SN+1]+\mathbf{b}_o(v))}{\partial x_u[t-SN+1]} \\
&= (1-\varphi^2(i_v^o[t-SN+1] \cdot z_v^o[t-SN+1])) \cdot o_v^o[t-SN+1] \cdot \sigma(i_v^i[t-SN+1]+\mathbf{b}_i(v)) \\
&\cdot (1-\sigma(i_v^i[t-SN+1]+\mathbf{b}_i(v)) \cdot \mathbf{W}_i(V+a,v) \cdot z_v^o[t-SN+1] \\
&+ (1-\varphi^2(i_v^o[t-SN+1] \cdot z_v^o[t-SN+1])) \cdot o_v^o[t-SN+1] \cdot i_v^o[t-SN+1] \cdot (1-\varphi^2(z_v^i[t-SN+1]+\mathbf{b}_c(v))) \\
&\cdot \mathbf{W}_c(V+a,v) + \varphi(i_v^o[t-SN+1] \cdot z_v^o[t-SN+1]) \\
&\cdot \sigma(o_v^i[t-SN+1]+\mathbf{b}_i(v)) \cdot (1-\sigma(o_v^i[t-SN+1]+\mathbf{b}_i(v)) \cdot \mathbf{W}_o(v+a,v)
\end{aligned} \tag{B.4}
$$

Thus, we can conclude that $h_v[t-SN+1]$ and $Z_v[t-SN+1]$ are differentiable w.r.t. $x_u[t-SN+1]$. Similarly, we can prove that $h_v[t-SN+1]$ and $Z_v[t-SN+1]$ are differentiable w.r.t. $x_1[t-SN+1]$, ..., $x_U[t-SN+1]$.

So $h_v[t-SN+1]$ and $Z_v[t-SN+1]$ are continuous and differentiable functions w.r.t. $x_1[t-SN+1]$, ..., $x_U[t-SN+1]$. Similarly, we can prove that $h_1[t-SN+1]$, ..., $h_v[t-$

$SN+1]$, $C_1[t-SN+1]$, ..., $Z_v[t-SN+1]$ are all continuous and differentiable functions w.r.t. $x_1[t-SN+1]$, ..., $x_U[t-SN+1]$. Furthermore, since the values of $h_1[t-SN+1]$, ..., $h_v[t-SN+1]$, $C_1[t-SN+1]$, ..., $Z_v[t-SN+1]$ are not influenced by $x_1[t-SN+2]$, ..., $x_1[t]$, ..., $x_U[t-SN+2]$, ..., $x_U[t]$, they are also continuous and differentiable functions w.r.t. $x_1[t-SN+2]$, ..., $x_1[t]$, ..., $x_U[t-SN+2]$, ..., $x_U[t]$, whose partial derivatives w.r.t. $x_1[t-SN+2]$, ..., $x_1[t]$, ..., $x_U[t-SN+2]$, ..., $x_U[t]$ equal 0.

For an arbitrary element b in the output vector, we have:

$$
\begin{aligned}
&h_v[t-SN+2] = \varphi(Z_v[t-SN+1]\cdot f_v^o[t-SN+2] + i_v^o[t-SN+2]\cdot z_v^o[t-SN+2])\cdot o_v^o[t-SN+2] \\
&= \varphi(Z_v[t-SN+1]\cdot\sigma(f_v^i[t-SN+2]+\mathbf{b}_f(v)) + \sigma(i_v^i[t-SN+2]+\mathbf{b}_i(v))\cdot\varphi(z_v^i[t-SN+2]+\mathbf{b}_c(v))) \\
&\cdot\sigma(o_v^i[t-SN+2]+\mathbf{b}_o(v)) \\
&= \varphi(Z_v[t-SN+1]\cdot\sigma(\sum_{v^*=1}^{B} h_{v^*}[t-SN+1]\cdot\mathbf{W}_f(v^*,v) + \sum_{u^*=1}^{U} x_{u^*}[t-SN+2]\cdot\mathbf{W}_f(V+u^*,v)+\mathbf{b}_f(v)) \\
&+\sigma(\sum_{v^*=1}^{B} h_{v^*}[t-SN+1]\cdot\mathbf{W}_i(v^*,v) + \sum_{u^*=1}^{U} x_{u^*}[t-SN+2]\cdot\mathbf{W}_i(V+u^*,v)+\mathbf{b}_i(v)) \\
&\cdot\varphi(\sum_{v^*=1}^{B} h_{v^*}[t-SN+1]\cdot\mathbf{W}_c(v^*,v) + \sum_{u^*=1}^{U} x_{u^*}[t-SN+2]\cdot\mathbf{W}_c(V+u^*,v)+\mathbf{b}_c(v))) \\
&\cdot\sigma(\sum_{v^*=1}^{B} h_{v^*}[t-SN+1]\cdot\mathbf{W}_o(v^*,v) + \sum_{u^*=1}^{U} x_{u^*}[t-SN+2]\cdot\mathbf{W}_o(V+u^*,v)+\mathbf{b}_o(v))
\end{aligned}
\tag{B.5}
$$

$$
\begin{aligned}
&Z_v[t-SN+2] = Z_v[t-SN+1]\cdot f_v^o[t-SN+2] + i_v^o[t-SN+2]\cdot z_v^o[t-SN+2] \\
&= Z_v[t-SN+1]\cdot\sigma(f_v^i[t-SN+2]+\mathbf{b}_f(v)) + \sigma(i_v^i[t-SN+2]+\mathbf{b}_i(v))\cdot\varphi(z_v^i[t-SN+2]+\mathbf{b}_c(v)) \\
&= Z_v[t-SN+1]\cdot\sigma(\sum_{v^*=1}^{B} h_{v^*}[t-SN+1]\cdot\mathbf{W}_f(v^*,v) + \sum_{u^*=1}^{U} x_{u^*}[t-SN+2]\cdot\mathbf{W}_f(V+u^*,v)+\mathbf{b}_f(v)) \\
&+\sigma(\sum_{v^*=1}^{B} h_{v^*}[t-SN+1]\cdot\mathbf{W}_i(v^*,v) + \sum_{u^*=1}^{U} x_{u^*}[t-SN+2]\cdot\mathbf{W}_i(V+u^*,v) \\
&+\mathbf{b}_i(v))\cdot\varphi(\sum_{v^*=1}^{B} h_{v^*}[t-SN+1]\cdot\mathbf{W}_c(v^*,v) + \sum_{u^*=1}^{U} x_{u^*}[t-SN+2]\cdot\mathbf{W}_c(V+u^*,v)+\mathbf{b}_c(v))
\end{aligned}
\tag{B.6}
$$

Since $h_1[t-SN+1]$, ..., $h_v[t-SN+1]$, $C_1[t-SN+1]$, ..., $Z_v[t-SN+1]$ are all continuous functions w.r.t. $x_1[t-SN+1]$, ..., $x_1[t]$, ..., $x_U[t-SN+1]$, ..., $x_U[t]$, $h_v[t-SN+2]$ and $Z_v[t-SN+2]$ are continuous functions about $x_1[t-SN+1]$, ..., $x_1[t]$, ..., $x_U[t-SN+1]$, ..., $x_U[t]$. For an arbitrary element a in the LSTM block's input vector and an arbitrary

$t^* \in t-SN+1,...,t$, the partial derivatives of $Z_v[t-SN+2]$ and $h_v[t-SN+2]$ w.r.t. $x_u[t^*]$ can be calculated as:

$$
\begin{aligned}
&\tfrac{\partial Z_v[t-SN+2]}{\partial x_u[t^*]} = \tfrac{\partial Z_v[t-SN+1]}{\partial x_u[t^*]} \cdot f_v^o[t-SN+2] + Z_v[t-SN+1] \cdot \tfrac{\partial f_v^o[t-SN+2]}{\partial x_u[t^*]} \\
&+ \tfrac{\partial i_v^o[t-SN+2]}{\partial x_u[t^*]} \cdot z_v^o[t-SN+2] + i_v^o[t-SN+2] \cdot \tfrac{\partial z_v^o[t-SN+2]}{\partial x_u[t^*]} \\
&= \begin{cases}
\tfrac{\partial Z_v[t-SN+1]}{\partial x_u[t^*]} \cdot f_v^o[t-SN+2] + Z_v[t-SN+1] \cdot f_v^o[t-SN+2] \cdot (1-f_v^o[t-SN+2]) \\
\cdot \sum\limits_{v^*=1}^{B} \tfrac{\partial h_{v^*}[t-SN+1]}{\partial x_u[t^*]} \cdot \mathbf{W}_f(v^*,v) + z_v^o[t-SN+2] \cdot i_v^o[t-SN+2] \cdot (1-i_v^o[t-SN+2]) \\
\cdot \sum\limits_{v^*=1}^{B} \tfrac{\partial h_{v^*}[t-SN+1]}{\partial x_u[t^*]} \cdot \mathbf{W}_i(v^*,v) + i_v^o[t-SN+2] \cdot (1-z_v^o[t-SN+2] \cdot z_v^o[t-SN+2]) \\
\cdot \sum\limits_{v^*=1}^{B} \tfrac{\partial h_{v^*}[t-SN+1]}{\partial x_u[t^*]} \cdot \mathbf{W}_c(v^*,v), \quad t^* \neq t-SN+1 \\
\tfrac{\partial Z_v[t-SN+1]}{\partial x_u[t^*]} \cdot f_v^o[t-SN+2] + Z_v[t-SN+1] \cdot f_v^o[t-SN+2] \cdot (1-f_v^o[t-SN+2]) \\
\cdot \mathbf{W}_f(V+t^*-t+SN,v) + z_v^o[t-SN+2] \cdot i_v^o[t-SN+2] \cdot (1-i_v^o[t-SN+2]) \\
\cdot \mathbf{W}_i(V+t^*-t+SN,v) + i_v^o[t-SN+2] \\
\cdot (1-z_v^o[t-SN+2] \cdot z_v^o[t-SN+2]) \cdot \mathbf{W}_c(V+t^*-t+SN,v), \quad t^* = t-SN+1
\end{cases}
\end{aligned}
\tag{B.7}
$$

$$
\begin{aligned}
&\tfrac{\partial h_v[t-SN+2]}{\partial x_u[t^*]} = \tfrac{\partial \varphi(Z_v[t-SN+2])}{\partial x_u[t^*]} \cdot o_v^o[t-SN+2] + \varphi(Z_v[t-SN+2]) \cdot \tfrac{\partial o_v^o[t-SN+2]}{\partial x_u[t^*]} \\
&= \begin{cases}
(1-\varphi^2(Z_v[t-SN+2])) \cdot \tfrac{\partial Z_v[t-SN+2]}{\partial x_u[t^*]} \cdot o_v^o[t-SN+2] + \varphi(Z_v[t-SN+2]) \cdot o_v^o[t-SN+2] \\
\cdot (1-o_v^o[t-SN+2]) \cdot \sum\limits_{v^*=1}^{B} \tfrac{\partial h_{v^*}[t-SN+1]}{\partial x_u[t^*]} \cdot \mathbf{W}_o(v^*,v), \; t^* \neq t-SN+1 \\
(1-\varphi^2(Z_v[t-SN+2])) \cdot \tfrac{\partial Z_v[t-SN+2]}{\partial x_u[t^*]} \cdot o_v^o[t-SN+2] \\
+\varphi(Z_v[t-SN+2]) \cdot o_v^o[t-SN+2] \cdot (1-o_v^o[t-SN+2]) \cdot \mathbf{W}_o(V+t^*-t+SN,v), \; t^* = t-SN+1
\end{cases}
\end{aligned}
\tag{B.8}
$$

Thus, $h_v[t-SN+2]$ and $Z_v[t-SN+2]$ are continuous and differentiable functions about $x_1[t-SN+1]$, ..., $x_1[t]$, ..., $x_U[t-SN+1]$, ..., $x_U[t]$. Furthermore, we can prove $h_1[t-SN+2]$, ..., $h_v[t-SN+2]$, $Z_1[t-SN+2]$, ..., $Z_v[t-SN+2]$ are all continuous and differentiable functions about $x_1[t-SN+1]$, ..., $x_1[t]$, ..., $x_U[t-SN+1]$, ..., $x_U[t]$.

Similar with $t-SN+2$, for any $t^* \in t-SN+3,...,t$, we can prove that $h_1[t^*]$, ..., $h_v[t^*]$, $Z_1[t^*]$, ..., $Z_v[t^*]$ are all continuous and differentiable functions about $x_1[t-SN+1]$, ..., $x_1[t]$,

..., $x_U[t-SN+1]$, ..., $x_U[t]$ given that $h_1[t^*-1]$, ..., $h_v[t^*-1]$, $Z_1[t^*-1]$, ..., $Z_v[t^*-1]$ are continuous and differentiable functions about $x_1[t-SN+1]$, ..., $x_1[t]$, ..., $x_U[t-SN+1]$, ..., $x_U[t]$. We arrive at **Lemma 2**.

Appendix C

Proof of Proposition 1

We adopt the proving method of mathematical induction. We use $h^m[t] = h_1^m[t], ..., h_{V^m}^m$ to denote the output vector of the M-th layer LSTM block, which has the scale of V^m. Obviously, $h^W[t] = y[t] = (y_1[t], ..., y_V[t])$.

For $m = 1$, according to **Lemma 2**, we have $h_1^1[t]$, ..., $h_1^1[t-SN+1]$, ..., $h_{V^1}^1[t]$, ..., $h_{V^1}^1[t-SN+1]$ are continuous and differentiable functions w.r.t. $x_1[t-SN+1]$, ..., $x_1[t]$, ..., $x_U[t-SN+1]$, ..., $x_U[t]$.

If we suppose for $m = m^*$, $h_1^{m^*}[t]$, ..., $h_1^{m^*}[t-SN+1]$, ..., $h_{V^{m^*}}^{m^*}[t]$, ..., $h_{V^{m^*}}^{m^*}[t-SN+1]$ are continuous and differentiable functions w.r.t. $x_1[t-SN+1]$, ..., $x_1[t]$, ..., $x_U[t-SN+1]$, ..., $x_U[t]$, $h_1^{w^*+1}[t]$, ..., $h_1^{m^*+1}[t-SN+1]$, ..., $h_{V^{m^*+1}}^{m^*+1}[t]$, ..., $h_{V^{m^*+1}}^{m^*+1}[t-SN+1]$ will also be continuous and differentiable functions w.r.t. $x_1[t-SN+1]$, ..., $x_1[t]$, ..., $x_U[t-SN+1]$, ..., $x_U[t]$ since $h_1^{m^*+1}[t]$, ..., $h_1^{m^*+1}[t-SN+1]$, ..., $h_{V^{m^*+1}}^{m^*+1}[t]$, ..., $h_{V^{m^*+1}}^{m^*+1}[t-SN+1]$ are continuous and differentiable functions w.r.t. $h_1^{m^*}[t]$, ..., $h_1^{m^*}[t-SN+1]$, ..., $h_{V^{m^*}}^{m^*}[t]$, ..., $h_{V^{m^*}}^{m^*}[t-SN+1]$ according to **Lemma 2**.

Therefore, can conclude that $h_1^m[t]$, ..., $h_1^m[t-SN+1]$, ..., $h_{V^m}^m[t]$, ..., $h_{V^m}^m[t-SN+1]$ are continuous and differentiable functions w.r.t. $x_1[t-SN+1]$, ..., $x_1[t]$, ..., $x_U[t-SN+1]$, ..., $x_U[t]$ for $\forall m \in Z^+$.

We arrive at **Proposition 1**.

Appendix D

Proof of Proposition 2

From **Lemma 1**, we have $|l_1[t] - l_2[t]| \leq \sqrt{k} \cdot \sigma + 2 \cdot \eta$ for $\forall t \in Z$.

Furthermore, according to **Proposition 1**, we have $y[t]$ is a continuous and differentiable function w.r.t. $l_1[t]$, ..., $l_1[t - SN + 1]$. $\frac{\partial y[t]}{\partial l_1[t]}$, ..., $\frac{\partial y[t]}{\partial l_1[t-SN+1]}$ are the partial derivatives.

Since σ and η are small enough and $|l_1[t] - l_2[t]|$, ..., $|l_1[t - SN + 1] - l_2[t - SN + 1]|$ are bounded by $\sqrt{k} \cdot \sigma + 2 \cdot \eta$, we have:

$$\begin{aligned} &|y[t] - \hat{y}'[t]| \leq |l_1[t] - l_2[t]| \cdot \frac{\partial y[t]}{\partial l_1[t]} + ... + |l_1[t - SN + 1] - l_2[t - SN + 1]| \cdot \frac{\partial y[t]}{\partial l_1[t-SN+1]} \\ &\leq (\sqrt{k} \cdot \sigma + 2 \cdot \eta) \cdot (\frac{\partial y[t]}{\partial l_1[t]} + ... + \frac{\partial y[t]}{\partial l_1[t-SN+1]}) \end{aligned} \tag{D.1}$$

Considering $y[t] = l_1[t + 1]$ and $|l_1[t + 1] - l_2[t + 1]| \leq \sqrt{k} \cdot \sigma + 2 \cdot \eta$, we have:

$$\begin{aligned} &|\hat{y}'[t] - l_2[t + 1]| \leq |y[t] - \hat{y}'[t]| + |y[t] - l_2[t + 1]| \\ &\leq |y[t] - \hat{y}'[t]| + |l_1[t + 1] - l_2[t + 1]| \\ &\leq (\sqrt{k} \cdot \sigma + 2 \cdot \eta) \cdot (1 + \frac{\partial y[t]}{\partial l_1[t]} + ... + \frac{\partial y[t]}{\partial l_1[t-SN+1]}) \end{aligned} \tag{D.2}$$

We arrive at **Proposition 2**.