Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Electrical, Electronics and Communications Engineering
(36^{th}cycle)

Data-Driven Mobile Traffic Modelling in Cellular Networks

By

Shuyang Li

******

Supervisor(s):
Prof. Enrico Magli, Supervisor

Doctoral Examination Committee:
Prof. Matteo Cesana, Politecnico di Milano
Prof. Gianluigi Ferrari, University of Parma
Prof. Tiziano Bianchi, Politecnico di Torino
Prof. Michela Meo, Politecnico di Torino
Prof. Attilio Fiandrotti, University of Torino

Politecnico di Torino
2024

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Shuyang Li
2024

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

I would like to dedicate this thesis to my beloved parents and girlfriend.
Their unwavering love and support have shaped me into the person I am today.

Acknowledgements

First and foremost, I would like to thank my supervisor Enrico Magli, who continuously guides and provides the support I need during my PhD study. His patience and encouragement helped me through the hardest time in this long journey, I am sure he is the dream mentor for any young PhD student. I also would like to thank Gianluca Francini and Giorgio Ghinamo for their great support; whenever I need advice from domain experts, they are always there. Finally, a warm thank you to Emanuele Caimotti for his support regarding workstation management.

This work was carried out in the Joint Open Lab and supported by the PhD research program of TIM S.p.A (Italy).

Abstract

Mobile traffic modelling is crucial for optimizing the network configuration and improving mobile user experience; by reliably modelling mobile traffic, it is feasible to obtain domain-specific knowledge and accurate mobile traffic predictions, which can help mobile operators better configure the network and adapt to the trend of mobile demand. In recent years, we have observed a dramatic growth in mobile traffic worldwide, and that raises more challenges for mobile operators to manage mobile networks efficiently. In this thesis, we explore how to model the characteristics of mobile network key performance indicators using solutions built on machine learning and deep learning.

For mobile traffic modelling, the problems can be generally subdivided into two subjects: how to obtain useful knowledge by analyzing mobile key performance indicators and how to predict them accurately. In this thesis, we first study how to recognize meaningful patterns which could be used to achieve better network management. Chapter 3 presents a novel time series clustering algorithm, this algorithm is responsible for creating clusters based on the variability of mobile traffic time series. For a cell in the mobile network, the potential activeness of its served mobile users can be reflected by the changing strength of the monitoring mobile traffic; hence the developed algorithm can be used to evaluate the importance level of each cell. Experiments have shown that the proposed clustering algorithm can identify different importance levels efficiently while having a fast running speed.

Chapter 4 and Chapter 5 focus on the problem of mobile traffic forecasting, where Chapter 4 explores the methods of predicting mobile traffic peaks and Chapter 5 focuses on general mobile traffic forecasting considering deployment constraints. In Chapter 4, we propose two deep learning-based mobile traffic peak predictors which can handle the trade-off between peak forecasting performance and general forecasting performance. Compared to widely used baseline approaches, these pre-

dictors have improved the peak forecasting performance significantly. Moreover, the proposed predictors are employed to implement a predictive detection approach, and this approach can efficiently predict if mobile traffic will be distributed among neighbouring cells in an imbalanced way, supporting the decision-making regarding the network management. Finally, Chapter 5 proposes two forecasting models respecting the deployment constraints which could be met under industrial scenarios, where memory and training time constraints are the main concerns of mobile operators. Extensive experiments are conducted on the real-world dataset, and the designed models can make accurate mobile traffic predictions having either few parameters or rather short training time.

To sum up this thesis, we study how to perform mobile traffic forecasting and knowledge extraction using machine learning and deep learning approaches. By evaluating the proposed methods on real-world industrial datasets, we prove that our approaches can be used to address different issues encountered under real scenarios, allowing mobile operators to manage and configure mobile networks more reliably and smartly.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, we have seen a dramatic growth of mobile traffic which is more rapid than ever; benefiting from the development of telecommunication technologies, the Internet is ubiquitous and connects people worldwide, permanently changing how we live [1]. Due to the increment of mobile devices and the rise of video streaming services over them, mobile data traffic volume is estimated to increase by more than twice in the period 2023–2027, and the mobile video traffic is predicted to grow by almost 30% annually through 2027 [2]; this massive growth of mobile demand makes the management of mobile networks more critical and challenging, especially for mobile network operators. To maintain the efficiency of mobile networks, the network configuration has to be updated over time to match the needs of mobile users; as many new technologies and architectures have been integrated with the telecommunication system such as the 5^{th} generation of mobile systems (5G) [3, 4] and others, the mobile network architectures are becoming much more complex and difficult to handle.

To meet the strict quality of service (QoS) requirements of different mobile services, it is essential to build an intelligent management system which can manage the networks automatically in an efficient way. Nowadays, data-driven methods have drawn research interests in this field, where machine learning (ML) and deep learning (DL) approaches are seen as promising techniques to enhance the ability of the mobile network to configure, optimize and heal itself [5–7]. ML and DL algorithms have shown great superiority at extracting hidden knowledge and recognizing various patterns from data input, making them fit mobile network applications very well;

every day mobile users generate a rather large amount of mobile data, and the mobile traffic activity is analyzed and recorded by mobile operators through generating different mobile network key performance indicators (KPIs) and logs. Considering that human activities and social events would significantly affect the mobile demand of base stations, meaning that different traffic usage patterns and network events are related to different user behaviours, which are affected by user habits and the city functionalities of the located area of base stations [8–11]; for example, the base stations located around the subway stations and bus stations usually carry much higher mobile traffic in commuting time, and the base stations located in the city centre are generally busy all day, which implies that mobile traffic is distributed across spatial and temporal domains in an inhomogeneous way. For mobile operators, unlocking the intricacies of traffic patterns concealed within mobile network KPIs is essential for the efficient operation and configuration of networks; this valuable information serves as a reflection of past usage patterns, guiding strategic decisions and optimizations. Nevertheless, the modelling of mobile traffic time series poses a great challenge due to the intricate nature of mobile demand, as the mobile traffic exhibits clear periodicity while retaining a certain level of randomness. In this case, the conventional statistic-based methods are outperformed by ML and DL approaches as the previous methods cannot model the complex non-linear relationships very well [12–15]; based on this reason, a growing number of research brings artificial intelligence (AI) to mobile network applications which is not limited to user mobility analysis [16, 17], network security [18–20], mobile data analysis [21–23], and network control [24, 25].

This thesis has been developed with the cooperation of Telecom Italia S.p.A, which is the largest telecommunication service provider in Italy expressed the interest and need for algorithms that could be implemented to explore mobile data efficiently. In this thesis, we explore mobile data analysis and modelling using data-driven solutions, facilitating enhanced knowledge extraction and predictions for cellular networks. Our work introduces novel machine learning and DL-based approaches to effectively address real-world challenges encountered in industrial scenarios.

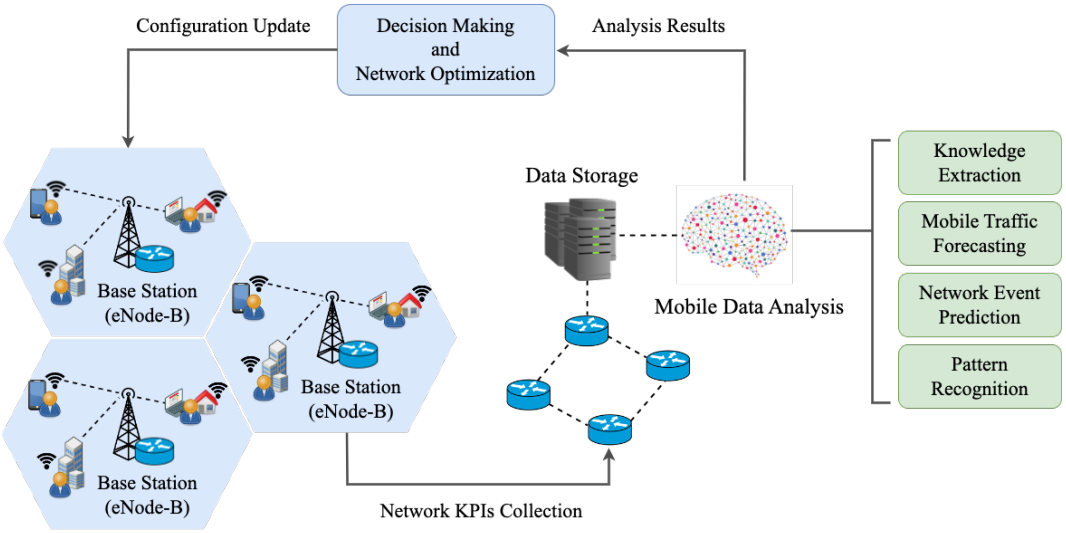


Fig. 1.1 An illustration of the process involving data collection, analysis, and optimization within mobile networks.

1.1 Management of mobile networks

To improve the QoS and user experience, the base station configuration must be configured frequently to adapt to the mobile demand of the covered areas. To optimize the network configuration more efficiently, a critical problem is how to perform data mining to maximize the value of the recorded mobile data. In mobile networks, the telecommunication infrastructures generate the network-level data stream continuously which describes the cellular network performance from a global point of view (e.g., downlink usage, average number of connected users, handover rate, throughput, etc.); the data reflects the comprehensive mobile user activities within the covered areas and exhibits strong temporal patterns due to the diurnal nature of human behaviour. Thanks to the existing spatiotemporal patterns of mobile traffic, it is possible to extract knowledge and discover the mobile data characteristics from the records through well-designed algorithms, and this topic has gained a lot of attention from mobile operators in recent years as the huge amount of stored data allows them to manage the networks in a more advanced way [5, 26, 27].

Network optimization is a rather challenging task which needs good cooperation between data collection, data storage, data analysis, optimization and configuration update of network infrastructures, and the cost of building the whole process is very high where many aspects within it require specific domain knowledge. To conduct

the network diagnosis and management automatically, mobile operators develop a complex system which is shown in Figure. 1.1. The illustration shows the general architecture of a management system of LTE mobile networks, and the system is composed of several major subcomponents. The radio access network is an important part of the system as it provides the mobile network KPIs for further analysis and gets the configuration update directly from the optimization module; within the radio access network, the cells are seen as the most fundamental component of it. The radio access network is composed of elements called evolved Node B (eNodeB) or E-UTRAN Node B; eNodeB is responsible for connecting mobile devices within its coverage to the mobile network [28, 29]. Each eNodeB consists of multiple cells, which act as connected targets for user devices; mobile operators can update the configuration of cells constantly to characterize a mobile network. Along with the running of cells, the observed mobile data is aggregated with a predefined interval (e.g., 10-minute, 15-minute, etc.) to create different mobile network KPIs, and each network KPI monitors a specific indicator related to the network performance. For a detailed understanding of mobile demand, the attention is focused on the KPIs to obtain insights into cell behaviour, leading to the need to analyze the time series of various KPIs. Once the mobile network KPIs are generated, the data storage system collects the KPIs from the radio access network, and this information is stored and fed into the mobile data analysis system. In mobile data analysis systems, ML and DL solutions are widely applied because of their efficiency and non-labor-intensive characteristics; based on the needs of different tasks, algorithms fall into categories such as supervised learning, unsupervised learning, or reinforcement learning. Once the analysis has been finished, the results are fed into the decision-making and network optimization system to determine the strategies for operating and configuration updates.

The mobile data analysis system plays a key role in the decision-making and optimization process, where the collected mobile KPIs are analyzed by AI approaches; the conducted analysis can be roughly subdivided into four categories: knowledge extraction, mobile traffic forecasting, network event prediction and pattern recognition. Knowledge extraction and pattern recognition are centred on extracting meaningful information and identifying regularities in data. While the results might not be directly implemented in the decision-making stage, they contribute to refining existing knowledge and cultivating a more profound comprehension of underlying relationships [30–32]. In contrast to the extensive exploration in mobile traffic fore-

casting and network event prediction, the research in mobile knowledge extraction and pattern recognition fields has not been thoroughly investigated; in this thesis, knowledge extraction and pattern recognition are discussed in Chapter 3 where we propose a novel clustering algorithm to distinguish different traffic usage patterns in city-wide mobile networks. Unlike these subjects, mobile traffic forecasting and network event prediction have garnered significant research attention over the years because the predictions can be easily integrated with the methods applied in network management and resource allocation [23, 33–35]. Mobile traffic forecasting aims to predict the mobile demand for the next few steps, and the predictions can be used to support configuration optimization or network event prediction. The prediction of network events can be made through either a direct or non-direct approach. In the former, predictions are made directly using observed network data, while the latter relies on other network analyses, such as short-term mobile traffic forecasting, to inform event predictions. In this thesis, we have conducted an intensive study of mobile traffic modelling and forecasting, taking into account the diverse requirements of various applications and deployment scenarios. In Chapter 4, we propose DL-based forecasting models which can significantly improve the prediction of mobile traffic peaks which have been rarely studied; the predictions are further used as the input of threshold-based algorithms to predict unobserved network events. In Chapter 5, two forecasting models are proposed to address diverse deployment constraints, including computational costs and storage restrictions, and this dives into the insights of mobile traffic modelling.

1.2 Thesis outline

In this section, we outline the thesis structure, chapter contents and the main contributions of each chapter.

Chapter 2 outlines the previous works related to time series analysis and mobile traffic modelling, providing minimal background knowledge of these topics to understand the contribution of the thesis.

Chapter 3 presents a novel time series clustering to analyze the cell behaviour in city-wide mobile networks. A novel clustering approach is proposed to create clusters based on the variability of the network KPI time series, which can be used to describe the importance level of each cell of mobile networks.

Chapter 4 presents solutions for predicting mobile traffic and the imbalanced distribution of mobile demand. In this chapter, we propose two DL models to improve the traffic peak forecasting performance. Besides predicting the mobile traffic peaks, we also propose a non-direct prediction approach to predict if the mobile traffic distributes in an imbalanced way among cells, whose occurrence is closely related to the observations of mobile traffic peaks.

Finally, Chapter 5 presents two novel DL-based mobile traffic forecasting models, and each one is proposed to address a certain issue which would be met during real-world deployment. These models are carefully designed to overcome the computational and storage restrictions while making accurate mobile traffic forecasting, allowing mobile operators to deploy forecasting models more flexibly based on the needs of applications.

1.3 List of publications

In this section, we outline the papers published and submitted during the PhD which are described in the thesis:

1. **Temporal dynamics clustering for analyzing cell behavior in mobile networks**, Li, Shuyang; Francini, Gianluca; Magli, Enrico, (2023) In: *Computer Networks*, Volume 223, March 2023, pages: 109578.

2. **To be Conservative or to be Aggressive? A Risk-Adaptive Mixture of Experts for Mobile Traffic Forecasting**, Li, Shuyang; Magli, Enrico; Francini, Gianluca, (2023) In: *ICC 2023 - IEEE International Conference on Communications*.

3. **Deep Learning Based Prediction of Traffic Peaks in Mobile Networks**, Li, Shuyang; Magli, Enrico; Francini, Gianluca; Ghinamo, Giorgio, (2023) In: *Computer Networks*, Volume 240, February 2024, pages: 110167.

4. **TDANet: An Efficient Solution For Short-Term Mobile Traffic Forecasting**, Li, Shuyang; Magli, Enrico; Francini, Gianluca, (2023) In: *IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*.

5. **Exploring Time Series Variability: A Training-Efficient Mobile Traffic Predictor**, Li, Shuyang; Magli, Enrico; Francini, Gianluca, (2024) Accepted by: *WCNC 2024 - IEEE Wireless Communications and Networking Conference*.

I am the first and corresponding author for all the publications described in this thesis. In particular, I independently devised all steps - from problem formulation to model design and results analysis; I also had the leading role in writing all of the papers.

Chapter 2

Related work

As discussed in Chapter 1, the radio access network consists of a large number of eNodeB and each eNodeB monitors the network states over time; to make the network management and operation more efficient, it is crucial to understand how to handle the collected KPI time series correctly. In this chapter, we present a literature review regarding the topics of time series analysis and mobile traffic modelling. Considering that many approaches employed in mobile traffic modelling are the same as or originate from the methods applied in time series analysis, we first present the previous works developed in the time series modelling field to provide a big picture, and then discuss the mobile network-specific modelling works. Furthermore, we highlight the difference between the current works and our works presented in different chapters of the thesis.

2.1 Time series analysis

Time series analysis is a very important topic and has historically been a key area of research in many different domains; time series data is everywhere in our lives, a complex industrial system, a vehicle or even a tiny medical device is generating time series data all the time. This data stream constantly monitors the states of the device or the system, which makes great value in strategic decision-making and predictive maintenance [36, 37]; for the applications in various domains, how to model time series efficiently to benefit from it has been a long-term research topic such as stock prediction [38, 39], climate modelling [40], power system management [41] and

arrhythmia diagnosis [42]. The subject of time series modelling is extensive and the objective of the works mainly belongs to the five categories:

- Generate time series and improve the data quality.
- Recognize specific temporal patterns from time series.
- Assign the time series to different groups based on their characteristics.
- Predict the future values of the time series.
- Compress time series to obtain better representations or overcome storage constraints.

Each category can be further subdivided into more specific topics concerning different needs of applications; for example, the creation of time series groups can be done either in the supervised way (time series classification) or in the unsupervised way (time series clustering), where the former task aims to assign labels to the sequences while the latter one focuses on finding similarities among the series without prior knowledge. In this section, we mainly review the literature in time series forecasting and clustering fields as the works presented in this thesis focus on mobile traffic prediction and knowledge extraction.

2.1.1 Time series forecasting

Time series forecasting has been extensively researched and has garnered significant attention since the previous century, and it is rather crucial to have accurate predictions for the applications requiring knowing future states in advance. The methods developed in this field can be divided into two categories: the traditional methods and the data-driven methods which are the approaches built on ML and DL algorithms. Traditional time series forecasting methods relying on the use of probability and statistics have achieved great success in the fields of finance, retail, industry and others. In the evolving era of big data, the continuous generation of massive, non-linear time series data adhering to multiple distribution patterns presents heightened challenges for time series forecasting methods. Consequently, practitioners leverage ML and DL approaches to address the intricacies of predicting highly complex time series data, yielding notable achievements in the process.

The traditional approaches mainly focus on parametric models informed by domain expertise, and they are built based on mathematical and statistical modelling [43]. Several classic prediction approaches are widely used including the autoregressive integrated moving average (ARIMA) [44, 45], simple exponential smoothing (SES) [46] and Holt-Winters methods [47]. ARIMA is proposed by Box and Jenkins which can be seen as one of the oldest time series forecasting approaches, this method was first used in the econometric field to predict indicators. The idea behind ARIMA is to model the relationship between the current observation and the previous ones, where the term "autoregressive" indicates that the ARIMA uses the relationship between an observation and the lagged observations from the previous time steps. ARIMA has been a standard method for time series forecasting for a long time and has been widely used in different domains [38, 48, 49]. Even though the ARIMA model is very popular in the forecasting field, it has some major drawbacks due to its characteristics. First, it is difficult for ARIMA to model the complex nonlinear relationships of time series limiting its performance for many industrial applications, and also ARIMA assumes that there is a constant standard deviation in errors and this may not be always satisfied from a practical point of view; lastly, the performance of ARIMA is very sensitive to parameter setting and the selection of the optimal values can be challenging which is dependent on the skill and experience of the forecasters [50, 51, 14]. Based on this reason, ARIMA has been proven as less effective in time series forecasting compared to neural networks and other data-driven methods [14, 52, 53]. SES and Holt-Winters methods are frequently discussed in tandem as there are similarities between their underlying mechanisms; SES is a simple method predicting future values using a weighted average of historical observations, and a smoothing parameter controls the weight assigned to the current sample; in the end, the final prediction is a weighted sum of the previous forecast and the most recent observation [54]. However, there is a significant drawback of SES: it cannot model complex temporal patterns, especially periodic patterns, and this critical issue often makes SES fail because many real-world time series consist of daily or weekly patterns. To overcome this problem, Holt-Winters methods have been proposed which are extensions of the traditional SES, and the biggest change is the new methods use additional terms to model the trend and seasonality within the time series. The introduction of the new terms allows Holt-Winters methods to make better predictions when the time series exhibits clear periodicity. Even though the statistical methods have been well-studied, they often fail when the time series is

non-stationary and not well-preprocessed following the requirements; the researchers also find that the statistical approaches cannot still model non-linear relationships making them fail in many real-world applications [55, 56]. The rise of data-driven methods has drawn attention and has established themselves as serious contenders to traditional statistical forecasting approaches, many works find that neural networks outperform the traditional methods and show superiority at learning the non-linearity, marking the onset of the era of DL approaches [57–60].

Neural networks are data-driven predictive models, their strong nonlinear approximation capabilities allow them to make accurate time series predictions after being trained to minimize a predefined loss function; another advantage of neural networks lies in their ability to automatically extract temporal patterns without relying on any specific theoretical assumptions about the data distribution, thereby minimizing the need for labour-intensive preprocessing. Through years of development, the neural network has become one of the most powerful tools for handling non-stationary time series and is widely employed in different domains [61–63]. For DL-based forecasting models, many advanced models have been built based on some fundamental architectures including multilayer perceptron (MLP), convolutional neural network (CNN) and recurrent neural network (RNN); the architecture design of the fundamental models differs a lot making them suitable for handling time series with different characteristics. In this case, how to select the proper architecture regarding the data's nature and the needs of applications is an essential topic; there is no actual "bad" model but to use the right model at the right time.

MLP is the most basic model of feed-forward neural networks, and its architecture only consists of three building blocks: an input layer, one or more hidden layers and an output layer [64], where the number of hidden layers measures the depth of this model. The layers of MLP are composed of trainable neurons, and the values of the parameters are found by solving an optimization problem. Unlike CNN applies the convolutional window and RNN introduces the recurrent connection, MLP has the simplest connection type and it can access all input data points at once. MLP is one of the first DL models which has been applied in the time series forecasting field because of its simple design but effective performance; many works find that MLP can make good predictions compared to the traditional methods [65–67]. Although MLP has been proven as an effective model, its design makes it unable to take into account the temporal order of time series samples, as MLP treats all samples equally and independently; the intra-series temporal dependencies are often ignored.

Therefore, the attention of researchers has been attracted by CNN and RNN because these models try to model temporal behaviour in different ways. For this reason, Nowadays MLP is widely used as a popular baseline approach or as a basic building block of more complex models.

CNN is a class of DL models designed for better processing the constructed grid data, such as images and video frames; its design makes it rather effective in computer vision applications. The core of its design is to employ a moving filter to perform convolution operation, eventually generating feature maps and modelling local patterns at different regions of the data; by stacking multiple convolutional layers, CNN can extract the high dimensional representation of raw data along with excellent visualization ability. Because time series can be seen as well-structured one-dimensional data points, CNN can be easily applied to time series data by modifying the size of applied filters. Compared to MLP, the convolutional layers used by CNN have shared weights, and this results in a much smaller model size. Benefitting from the characteristics called distortion invariant, CNN can extract local features reliably no matter where the features are in the raw data input, and that makes CNN a good candidate for predicting time series with specific temporal patterns [68–71]. Besides the conventional CNN, temporal convolutional network (TCN) is a more specialized version designed for addressing sequence modelling problems [72]. Compared to CNN, TCN is a more advanced model applying a convolution operation called dilated causal convolution. In dilated causal convolution, the output of convolution at a given time step depends only on the current and preceding data points and not on future samples, which respects the nature of time series; the dilated causal convolution also introduces gaps (dilations) between the elements of the filter, and this makes TCN having a larger receptive field and better capturing the long-term dependencies within the time series. These advantages make TCN a popular approach in the time series forecasting field [73, 74], but its performance depends on the characteristics of input data which could differ a lot on different datasets [56].

RNN is the last fundamental architecture presented in this section, which is designed specifically for the tasks related to sequence modelling [75]. As mentioned in the previous sections, MLP and CNN have their characteristics where MLP treats all input samples equally without considering the temporal dependencies within the time series and CNN focuses on extracting the local temporal patterns; unlike these architectures, RNN introduces a recurrent connection connecting the current time step to the preceding ones to model the evolving temporal patterns of time

series, which fits the time series nature very well and achieves good results [76]. However, a critical problem of RNN is that the model has difficulty modelling the long-term temporal dependencies, and it may meet the vanishing gradient problem when the sequence is long [77]. To address this issue, long short-term memory (LSTM) [78] and gated recurrent unit (GRU) [79] are proposed; by applying the gate mechanism, they can better capture the correlation among samples and address the vanishing gradient problem. The implementation of the gate mechanism greatly improves the sequence modelling ability, making LSTM and GRU the two most important members of the RNN family. In time series forecasting, LSTM and GRU are rather popular and obtain excellent results in many domains [80, 81]; they have also inspired many models which are designed especially for time series forecasting problems.

Recently, significant efforts have been dedicated to enhancing the time series forecasting performance using DL models, which leads to the burst of more advanced models built based on the mentioned fundamental architectures; these models are much more powerful than the conventional neural networks and have become the state-of-the-art methods on different datasets. In general, there are three classes of DL models concerning the model design: feed-forward, RNN-based, and Transformer-based models. Feed-forward models are the models built without using any RNN and Transformer component, and the architecture is usually composed of MLP, CNN and residual connections; so far the most widely used models under this category are DLinear [82], NLinear [82], N-BEATS [83] and TCN [72]. NLinear and DLinear are modified versions of MLP where NLinear performs normalization and denormalization before and after making predictions, and DLinear integrates the MLP with a decomposition module. N-BEATS is a more complex model compared to the previous ones; it runs a sequential analysis of the input signal recursively, every time it forecasts part of the predictions and removes the well-modelled part from the previous signal, which performs an ensemble style forecasting. For RNN-based models, LSTNet [84] and TPA-LSTM [85] are seen as state-of-the-art models for time series forecasting and they share similar design insights: both of them try to model the long-term and periodic patterns in a better way, where LSTNet employs multiple RNNs with skip-connections and TPA-LSTM relies on the attention mechanism. Besides the two models, DeepAR [86] and MQ-RNN [87] are also popular considering that they can make probabilistic predictions and quantile predictions, this satisfies the needs of the applications requiring to model the future distribution

of a random variable. The last category is the Transformer-based models which are popular nowadays; the design of these models employs the encoder-decoder architecture along with multi-head self-attention mechanisms. Transformer was originally proposed to address natural language processing tasks [88], and it is found as an effective solution to handle sequences. In this case, many Transformer-based models have been developed to address the long-term time series forecasting problem, and the most famous ones are Informer [89] and Autoformer [90]. Even though these models have been intensively studied in recent years, some works point out that the Transformer-based models may not be that effective in prediction compared to other simpler architectures [82, 91]. As we present in the section, the selection of a forecasting model mainly depends on the characteristics of time series, and it is extremely hard to find an architecture which can obtain good results on all datasets.

2.1.2 Time series clustering

Clustering analysis is a frequently used data mining technique which can split the input data into different groups (clusters) based on their relevant similarities; during this process, the similarities between an object and different clusters are calculated individually, and the object will be assigned to the most similar cluster in a pure data-driven fashion without having any specific knowledge of the group characteristics [92]. Time series clustering is a specialized domain within the broader field of clustering, and this topic focuses on clustering time series based on the intra-series or inter-series similarities [93]. Because time series clustering is done in an unsupervised way, that makes it a useful tool for exploratory data analysis, and serves as a preliminary step for various other data mining tasks or as an integral component within a complex system.

How to apply time series clustering efficiently requires a clear understanding of its procedure; we can understand the whole process from four different dimensions:

- Clustering level.
- Type of clustering method.
- Clustering algorithm.
- Similarity measure.

The difference between the second term and the third term is that the type of clustering method represents the general fashion of clustering workflow, and the clustering algorithm means the clustering algorithm actually employed in the clustering stage of the clustering analysis; more specifically, the clustering algorithm is only a stage of the clustering method, and the clustering method may also include other stages such as preprocessing or feature extraction. In this section, we will discuss the corresponding contents following the order of clustering level, cluster method, clustering algorithm and similarity measure.

Time series clustering can be done on three levels: point-level, subsequence-level and sequence-level [94]. The approach at the lowest level is point-level clustering whose aim is to split the samples of a time series into different groups, each group only consists of partial data points of the time series; this type of clustering is often used to isolate the outliers within the time series to detect anomalies or to be used as a denoising approach before applying time series forecasting and classification algorithms [95, 96, 28]. The subsequence-level clustering aims to perform clustering analysis based on the segments of time series; a set of subsequences is first created by using a sliding window on a single time series, and then a clustering algorithm is employed to find the similarity among them. This approach is usually used to classify different temporal patterns within a specific sequence and is widely used in many algorithms including rule discovery [97–99]. The last category is sequence-level clustering which clusters individual time series according to the inter-series similarity; this clustering fashion is the most popular one which has been applied in many works [100–102].

Besides the clustering level, there are three types of clustering methods from a general point of view which are the shape-based method, feature-based method and model-based method; the difference between these methods is how they use the raw time series to perform clustering analysis [93]. The shape-based method applies clustering algorithms directly (e.g., K-means, etc.) on raw time series, considering the stretching and contracting of the time axes to match the shapes of time series as closely as possible. The situation in the feature-based approach is different from the previous one, where a feature (new representation) is calculated from the raw time series, and then a conventional clustering algorithm is used to cluster the calculated feature vectors. Unlike the shape-based method which has many candidate distance functions, the feature-based method usually uses the the Euclidean distance to measure the similarity between time series objects [103]. The model-based method

is rarely used in the literature compared to the others, and this method converts the raw time series into model parameters; for example, this approach would build a parametric model for each time series individually. The employed clustering algorithm is used to measure the similarity between the generated parameters [104]. In the time series clustering field, the shape-based and feature-based methods are used in most cases.

Once we determine the clustering level and the type of clustering method, the remaining things are to choose proper clustering algorithms and distance measures which are the core of clustering analysis. For exploring time series patterns, the widely used groups of clustering algorithms are partitioning clustering, hierarchical clustering and density-based clustering. The partitioning clustering algorithms are popular both in academia and industry; when we decide the number of generated clusters, the algorithms split the sequences into different groups and each group is composed of at least one time series. The most famous partitioning algorithm is K-means [105] which calculates a centroid for each cluster to characterise the general behaviour of the cluster; the centroid is usually the mean value of all objects within the cluster, and a new object is assigned to a specific cluster only if the object is closest to that centroid. Another well-known partitioning algorithm is K-medoids (PAM) [106] which is very similar to K-means; the main difference between them is about the selection of the cluster center: in K-means the centroid is defined as the mean value of all objects within the cluster while K-medoids uses medoid as the cluster center and the medoid is the object minimizing the sum of distances to all other objects in the cluster. In this case, the centroid of K-means is obtained through calculating and may not be a real object, which is not the case in K-medoids. Compared to K-means, K-medoids is more robust to outliers because the selection of cluster center does not rely on calculations; at the same time, K-medoids is much more time-consuming as it requires more calculations to select the right medoids [107, 94]. The effectiveness and simpleness of K-means and K-medoids make them employed in many works [103, 108, 109]. Moreover, a more advanced version of K-medoids is proposed to support the data mining on spatial datasets, which is called CLARANS [107]. Both K-means and K-medoids create clusters in a “hard” way, meaning that each time series object can only belong to a specific cluster. However, the clustering analysis can also be performed in a fuzzy way by using the fuzzy c-means algorithm [110]; this algorithm can assign objects to clusters in a fuzzy way, and each object holds a membership vector storing the probabilities that this

object belongs to each cluster; in this case, the clustering can be done in a "soft" way. Hierarchical clustering focuses on creating clusters in an agglomerative or divisive manner; a hierarchy of clusters is built by it and this architecture is usually represented as a tree-like structure known as a dendrogram [106]. Agglomerative hierarchical clustering is a bottom-up approach which considers each object an individual cluster and merges all of them gradually until there is only one cluster; divisive hierarchical clustering does it in the opposite way which gradually splits a large cluster into smaller ones. Compared to K-means, hierarchical clustering is required to calculate the pairwise distance matrix, and this makes it much slower than K-means and not suitable for clustering analysis on large time series datasets [111]. Density-based clustering algorithms create clusters based on the density of objects in the feature space, and that allows it to construct clusters of arbitrary shapes. The most famous algorithm in this family is called DBSCAN [112], and this approach creates a cluster by expanding it only if the neighbour objects are dense; this process would end if there are not enough neighbours. One advantage of DBSCAN is there is no need to specify the number of clusters while the algorithm is capable of discovering clusters without prior information. Even though this approach is convenient, it is not a primary choice in time series clustering due to the high complexity [93].

For the similarity measure, we want to discuss the two most popular ones: Euclidean distance and dynamic time warping (DTW). Euclidean distance is usually used to measure the similarity between time series in time or the similarity between generated features; for the case of similarity in time, the similarity is calculated on each time step and that often leads to the evaluation of how the two sequences match in time. DTW [113] is well-known for its invariance against signal shifting [114]; to measure the similarity between sequences, DTW first computes a pairwise distance matrix between points of two time series, and then a dynamic program following a recursive manner is used to find the minimum-cost alignment between them [115]. In this way, DTW is usually used to find similar time series in shape. However, the complexity of DTW is rather high and that limits the usability of DTW on medium and large datasets.

2.2 Mobile traffic modelling

In the previous sections, we have reviewed the works related to time series modelling, and this section will present the works which have been done specifically for mobile networks. Mobile networks have become important infrastructures providing network services to mobile users around the city, and the increasingly complex network structure makes maintenance more challenging than ever. To satisfy the needs of different mobile applications, the configuration of telecommunication devices has to be updated remotely to adapt to the trend of mobile traffic usage. Powered by ML and DL techniques, predictive maintenance and resource allocation can be done effectively and mobile networks are becoming more smarter and reliable. For AI-powered mobile network applications, how to model mobile traffic to maximize the value of data is crucial, and this subject can be further divided into more specific topics where mobile traffic forecasting and knowledge extraction are important among them.

For mobile operators, it is always desired to obtain the updated cell configuration in time to prevent the effect of delayed observations; thus, accurate mobile traffic predictions are necessary for later optimization of configurations. More specifically, it would bring benefits to mobile operators if the short-term mobile traffic predictions could be made accurately based on historical observations [7]. To predict mobile traffic, many approaches mentioned in the section on time series forecasting have been used in this field, and we also observe the same technique evolution from statistical methods to DL models [116]. The representative employed statistical approaches consist of ARIMA [117, 118], SES [119] and Holt–Winters models [120]; Even though these models have much less computational cost which makes them good candidates in embedded devices with limited computation abilities, most of them can only model the linear relationship and this makes them being outperformed by DL approaches [121]. Compared with statistical models, DL-based mobile traffic forecasting is the general trend in this decade. Much research shows that even the fundamental neural networks can outperform the traditional statistical methods in mobile traffic prediction, such as MLP [122], GRU [123], LSTM [124–126] and CNN [127–129]. However, the ability to model non-linear relationships is only the basic requirement for predicting mobile traffic; although the fundamental DL models already outperform statistical models, they are still insufficient for predicting mobile traffic accurately. Due to the nature of mobile traffic, the time series is a

mixture of different patterns including the daily pattern, weekly pattern and recent trend, and makes the prediction exceptionally difficult as it is hard to quantize the contribution of each term. In this case, more advanced models are employed to predict mobile traffic which consists of and is not limited to Transformer-based [130, 131] and attention-based models [132–134]. Among these works, two issues are often ignored and have been rarely studied. First, none of these works investigates how to predict mobile traffic peaks in a better way, and peak prediction is a critical problem, especially for the configuration update of the cells with high mobile demand and many active users; the mismatch between configuration and mobile demand could result in low QoS or even the discontinuity of mobile service in the worst case. In this thesis, this issue has been discussed thoroughly in Chapter 4 where we propose specific models to improve the peak forecasting performance. Another issue is related to the model design of mobile traffic predictors; many works in the mobile traffic forecasting field develop complex neural networks to obtain better predictions, but the deployment difficulties which may be encountered in real cases are ignored in most cases. On the side of mobile operators, the storage and computational constraints are non-negligible factors during the model deployment; for example, mobile operators would periodically retrain the forecasting model to better capture the recent trend of mobile traffic, and the training could have a very high cost and could not be finished in limited time on large datasets if the forecasting model has low computational efficiency. In Chapter 5, we discuss in detail how to design the forecasting model in order to satisfy different deployment constraints.

Compared to mobile traffic forecasting, knowledge extraction is a less-discussed subject in the mobile network applications field. Recently, the management of mobile networks has become more challenging, especially when we lack the knowledge of usage patterns in complicated urban environments. To gain a deeper insight into mobile networks, extensive studies have been done to analyze the behaviour of mobile networks. The performed analysis can be divided into two broad categories: (1) study of mobile user behaviour and consumption of different mobile services and (2) study of mobile network behaviour on the side of network operators. For user behaviour analysis, [135] proposes a framework to detect anomalous telecom customer behaviour and analyze users' usage patterns, to provide supplemental information for precision marketing of telecom operators and preventing criminal behaviours. [11, 136] perform national scale analysis of user usage patterns. [136] focuses on the data usage patterns of geographically diverse mobile users meanwhile

[11] studies the difference in mobile service consumption by considering various service categories, such as streaming, social networking, cloud service and gaming. For the studies on the network operator side, [137, 138] use K-means to detect mobile traffic anomalies by using call details record; these works focus on detecting outliers of each time series which aims to study the behaviour of a single record instead of the behaviour of sequence. Besides the clustering approach, [139] uses DL models to detect anomalies within traffic sequences, but their dataset is very small as it only consists of data collected from three base stations. [140] uses a huge dataset collected from Beijing to study the behaviour of cellular towers in different areas. Through clustering analysis, they find that the mobile traffic usage pattern is very different when the functionality of analyzed areas differs a lot; five functional areas are discussed, including residential area, transport area, office area, entertainment area and comprehensive area. Among the aforementioned studies, none of them explores how to evaluate the importance of cells within the network, which is valuable for resource allocation. As presented in [140], the mobile traffic usage pattern is different for the type of functional areas; however, the cells can also have very different behaviours even though they are located in the same area, and the importance level of each cell is strongly related to the potential activeness of users assigned to it. Under this scenario, Chapter 3 introduces a new clustering algorithm to evaluate the importance of cells in a finer granularity, allowing mobile operators to determine the potential optimization priority of each cell.

Chapter 3

Analysis of cell behaviour in mobile networks

As discussed in Chapter 2, the clustering analysis with a finer granularity would benefit mobile operators to allocate mobile resources more efficiently, and the explored patterns are used to enhance our knowledge affecting the decision-making of optimization. In the previous literature, the difference between mobile traffic usage patterns has been well-studied, and the results prove that area functionality is a key factor that should be considered for understanding the behaviour of cells. However, this knowledge is quite general with a coarse granularity; for mobile network management, it is insufficient to know only the difference between the located areas, understanding the pattern of individual cells is also very important. In this chapter, the material appeared in my publication [141].

3.1 Exploring the cell importance

The usage pattern of mobile networks is strongly related to human activities, which results in an inhomogeneous distribution of user demand. Previous research [140] shows that the usage pattern is very different when the functionality of areas differs a lot. For example, the cellular towers located in office areas usually suffer heavier mobile demand than others during working time. Even when two cellular towers are located in the same area, it does not mean they are equally important, and it is quite naive to allocate to them a similar amount of network resources. For

mobile operators, to manage mobile networks in an efficient way it is important to understand the behaviour of traffic patterns. To acquire a deeper understanding of mobile demand, the attention is focused on cells of LTE mobile networks, which can be seen as the most fundamental component of the access network layer. The access network layer is composed of eNodeB and eNodeB is responsible for connecting mobile devices within its coverage to the mobile network [137, 29]. Each eNodeB consists of multiple cells, and these cells act as connected targets for mobile devices. mobile operators can update the configuration of cells constantly to characterize a mobile network; under this scenario, exploring the traffic pattern of cells provides valuable information for maintaining the network. If we want to obtain insights into cell behaviour, it is essential to analyze the time series of data generated by cells. Based on what we mentioned previously, mobile demand is distributed in an inhomogeneous way among cells, which means some cells serve more mobile users than others. In this case, the more heavily loaded cells have higher priority in the management of the LTE network, and they should also be allocated more network resources. However, cell priority is not known in advance, hence the need for cell behaviour analysis techniques, which typically employ unsupervised learning methods.

Unsupervised learning is a preferred method for exploring unknown patterns within data. Among various techniques, clustering emerges as a highly utilized approach in exploratory data analysis. By employing cluster analysis, objects are divided into different groups according to their relative similarities, with each cluster containing objects that share similarities with one another. To identify the cell behaviour, cells have to be split into different groups based on their temporal behaviour, and the problem can be cast as time series clustering. Specifically, the time series associated with each cell may contain the temporal evolution of several parameters of interest related to the cell, which can be used to determine the importance of the cell itself. Parameters may include different network KPIs such as the number of connected users, downlink usage, handover and others. For time series clustering, many works have been done in various industrial domains, and most of them focus on the recognition of different waveforms and the identification of anomalies within a time series. For example, [137] applies K-means to detect anomalies within mobile traffic time series. [101] uses different clustering methods to study the temporal pattern of energy consumption. While waveforms and anomalies are crucial aspects of time series, they alone do not furnish sufficient information in our context. If

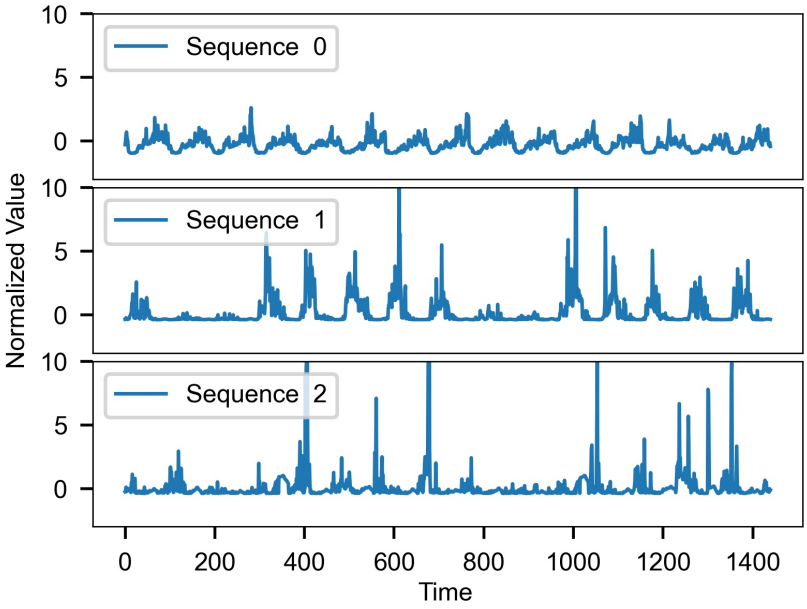


Fig. 3.1 Downlink usage sequences with different oscillation levels.

we want to know the importance of cells to determine their priority, it becomes essential to create clusters that take into account the variability of time series data. In mobile network scenarios, if a cell covers plenty of mobile devices, spikes and strong oscillations can be observed in its time series, which exhibits strong temporal dynamics. Consequently, it is anticipated that a time series clustering algorithm should cluster series based on their level of temporal dynamics, since this is a very useful indicator of the importance of the cell. This requires to 1) identify appropriate features capable of capturing dynamic behaviour, and 2) couple them with a suitable backbone clustering method.

In order to group time series based on their temporal dynamics, it is essential to correctly measure this characteristic. Due to the inherent vagueness in defining temporal dynamics, identifying a precise criterion for evaluation poses a challenge. Typically, a time series characterized by strong temporal dynamics exhibits spikes and irregularities devoid of a specific pattern. Conventional clustering approaches cannot be used to group time series by temporal dynamics because of the limitation of distance measures and computational cost. As presented in Chapter 2, Euclidean distance and DTW have been widely used to measure the similarity between sequences; compared to Euclidean distance, DTW does a better job at handling sequence mis-

alignment. However, the complexity of DTW is quadratic, which limits the usability of DTW on medium and large datasets. As shown in Figure 3.1, it is obvious that time series 1 is more similar to time series 2 than to time series 0 in a semantic sense by observing the sequences. But if we calculate the Euclidean distance between these series, the distance between series 1 and series 0 is 56.28 while the distance between series 1 and series 2 is 69.30. If we perform clustering with this similarity measure, series 0 and series 1 should be grouped into the same cluster which is undesired. Considering this perspective, Euclidean distance is not always a good option for comparing the temporal dynamics of time series. If the similarity is measured by DTW, in this case, series 1 is more similar to series 2, and this may lead to correct cluster creation. Nevertheless, DTW incurs a substantial computational cost, increasing quadratically with the length of the series. In mobile network scenarios, a time series is usually very long, and that makes DTW impractical in mobile time series clustering. The problem of computational cost becomes more critical, particularly when the algorithm requires the calculation of pairwise distances among all objects, such as K-medoids and hierarchical clustering. In this case, although a clustering algorithm theoretically can accurately assess the temporal dynamics of time series, the associated complexity may render the method impractical or unusable. Based on the above-mentioned issues, a clustering algorithm should address two challenging problems: (1) How to define features that allow to compare temporal dynamics between time series? (2) How to reduce computational cost while measuring similarity correctly?

To address these issues, we propose a new feature-based clustering approach called Temporal Dynamics Clustering (TDC). This algorithm measures the variability of time series by comparing the distribution of the first-order differences sequences, and computational cost is greatly reduced by an operation called distribution summarization. To evaluate the performance of our algorithm, the proposed method is compared to the baselines which are shape-based approaches. Unlike the feature-based approach which uses the extracted features of the original time series, the shape-based approach uses raw time series directly for clustering, and the features are the sequences of the considered variables [93]. The main contributions are summarized as follows:

- We propose a new time series clustering algorithm to group time series based on their temporal dynamics with low cost, where clusters are created based

on the degree of variability of time series. Our algorithm employs suitably defined features coupled with a conventional backbone clustering method; it is evaluated on real-world datasets collected from a mobile network, achieving remarkable performance compared to baselines. The generated clusters are better separated, and our algorithm is 100x faster than other clustering approaches, such as K-medoids and hierarchical clustering.

- We train neural networks on the clusters created by TDC, and the results show that the forecasting difficulty of these clusters is different, which provides supplemental information for evaluating and improving time series forecasting models.

- Using TDC, we analyze the behaviour of cells and obtain interpretable results. Our algorithm splits cells into three clusters, and several areas are identified as important by observing the geographical distribution of clusters. The identified areas play important roles in daily life, such as university, train station and historical district; this fills the gap of cell level analysis of mobile traffic.

This chapter is structured as follows. Section 3.2 presents the proposed algorithm, baselines and metrics for evaluating performance. In Section 3.3 we introduce three real-world datasets which are used to perform experiments. The proposed algorithm is compared with other baselines, and its behaviour is analyzed in detail. We also provide a real usage case of our method which explores cell behaviour in Turin, Italy. In Section 3.4, we draw some conclusions.

3.2 Clustering analysis

3.2.1 TDC

As shown in Figure 3.2, TDC consists of three parts: generation of first-order differences sequence, distribution summarization and K-means clustering algorithm. The first two steps act as the feature extractor which is used to generate features for the subsequent clustering process; the feature extractor of conventional approaches is usually very simple, as these approaches use the samples of variables directly as the features for clustering. Unlike the conventional approaches, in TDC features are

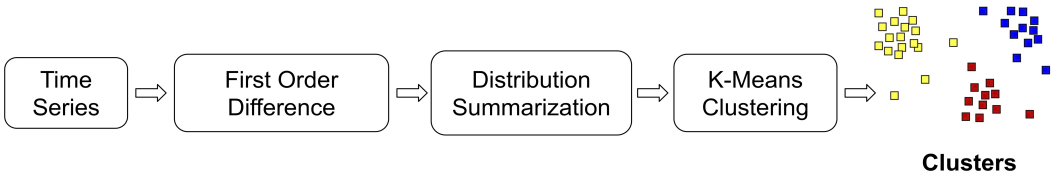


Fig. 3.2 Workflow of TDC.

designed to summarize the dynamic behaviour of the time series, and are then fed into the backbone algorithm (clustering engine) to create clusters.

The aim of the first step is to measure the variation between adjacent elements; instead of using the original time series, the first-order difference is calculated for the series of each variable. The mathematical expression of the first-order difference is the following:

$$\Delta y_t = |y_t - y_{t-1}|, \tag{3.1}$$

where Δy_t is the absolute value of the difference at time t, and y_t and y_{t-1} are the values of the time series at time t and $t-1$. In this way, for each variable we convert the original time series into a sequence of differences, which provides a more natural way to measure local temporal dynamics.

The second step is called distribution summarization, which aims at summarizing the characteristics of the time series in a way that enables analysis of dynamics with a small computational cost and retains the general behaviour of data distribution; to make the algorithm practical on industrial datasets, this process should be less computationally expensive. For clustering time series by temporal dynamics, distribution summarization should highlight the highly varying parts of the differences sequence while retaining the global behaviour. Figure 3.3 illustrates the CDF (Cumulative Distribution Function) of the differences sequence of two important mobile network KPIs: the downlink usage and the average number of connected users. Observing the figures, we find the CDFs have similar behaviour when the quantile is not so high, and the main difference between CDFs relates to the behaviour of the tail. Long tail can be seen as an indicator of strong oscillation considering these CDFs are plotted based on the differences sequences instead of the original series, which means the tail behaviour encodes the "rare event" of each series. If we want to describe this distribution correctly, it is essential to summarize the tail behaviour. However, this

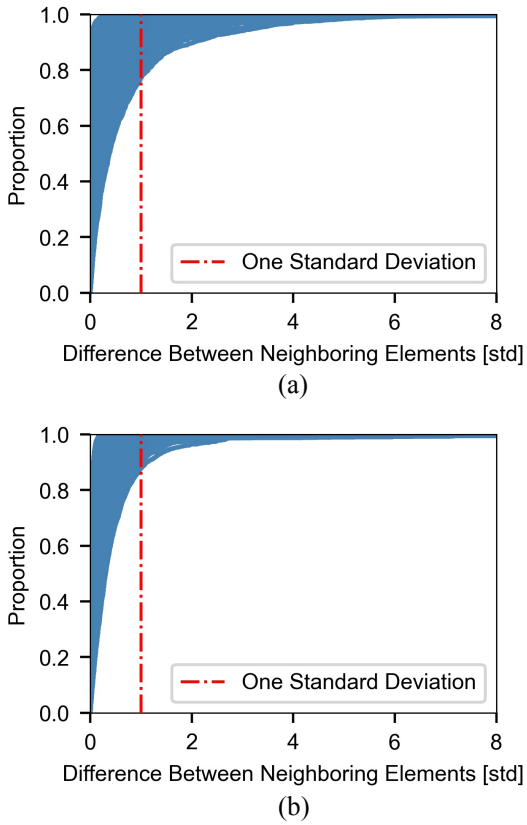


Fig. 3.3 Cumulative Distribution Function (CDF) of two mobile network KPIs. (a) Downlink usage; (b) Average number of connected users.

requires to define exactly what values belong to the tail. For addressing this issue, the proposed algorithm employs extreme value theory which is mainly used to estimate the tail distribution for analyzing rare events.

Extreme value theory studies the distribution of extreme realizations of a given distribution function, or of a stochastic process [142, 143]. It is widely used to study the behaviour of the distribution's tail, especially in financial and environmental fields. To estimate the tail distribution, it is essential to define the tail of the observed records. A popular technique in this field is Peaks-Over-Threshold, wherein a predefined threshold is used to identify the tail part. In TDC, we follow a data-dependent rule proposed by [144] which is also used in [145, 146]. Compared to other tail estimation methods, this approach does not require any manual check [147, 148] or to solve an optimization problem [149]. According to the rule, the

largest u elements of the differences sequence are defined as the tail part, and u is calculated based on the following formula:

$$u = \frac{(T-1)^{2/3}}{\log(\log(T-1))}, \tag{3.2}$$

where T is the length of original time series. Based on tail index u, a quantile index I_q is calculated to split the distribution into the main part and the tail part:

$$I_q = 1 - \frac{u}{T-1}, \tag{3.3}$$

$$\Delta Y^{main} = \{\Delta y_t \mid \forall t \in [1, T-1] \text{ and } \Delta y_t \leq Quantile(\Delta Y,\ I_q)\}, \tag{3.4}$$

$$\Delta Y^{tail} = \{\Delta y_t \mid \forall t \in [1, T-1] \text{ and } \Delta y_t > Quantile(\Delta Y,\ I_q)\}, \tag{3.5}$$

where $Quantile(\Delta Y,\ I_q)$ is the calculated quantile of differences sequence ΔY given quantile index I_q, and ΔY^{main} and ΔY^{tail} are the main part and the tail part of the series. After doing this, the distribution of time series is summarized as a new representation r:

$$r = [\ \mu^{main},\ \sigma^{main},\ \mu^{tail}\ ], \tag{3.6}$$

where μ^{main} is the mean value of ΔY^{main}, σ^{main} is the standard deviation of ΔY^{main} and μ^{tail} is the mean value of ΔY^{tail}. By summarizing the behaviour of different sequences, the algorithm generates a new representation for the original time series which has much fewer elements, leading to a great reduction in computational cost of the subsequent clustering step. If the time series is multivariate, the representation should be computed individually for each variable, and the dimension of generated representation is $r \in R^{3 \cdot V}$ where V is the number of variables.

In the last step of TDC, the generated representation serves as input for the clustering process. Obtaining meaningful results and performing efficient clustering in TDC relies on the careful choice of a clustering engine, and this involves choosing

Algorithm 1 TDC

Input: Time Series $Y = \{y_1, ..., y_N\},\ y_N \in R^{T,V}$. The number of clusters k.
Result: Cluster Partition $P = \{C_1, ..., C_k\}$.

$u = \frac{(T-1)^{\frac{2}{3}}}{\log(\log(T-1))}$ ▷ Tail Index
$I_q = 1 - \frac{u}{T-1}$ ▷ Quantile Index
$R = [\,]$ ▷ Empty Representation

for Object $n = 1$ to N **do**
  $r = [\,]$
  for Variable $v = 1$ to V **do**
    S = Calculate First-Order Difference(y_n^v)
    Q = Calculate Quantile(S, I_q)
    $S^{main} = \{s_h : s_h \leq Q \text{ for } h = 1, ..., T-1\}$
    $S^{tail} = \{s_h : s_h > Q \text{ for } h = 1, ..., T-1\}$
    $\mu^{main} = Mean(S^{main})$
    $\sigma^{main} = Std(S^{main})$
    $\mu^{tail} = Mean(S^{tail})$
    append μ^{main}, σ^{main}, μ^{tail} to r
  end for
  append r to R
end for

P = K-means($R,\ k$) ▷ Run K-means with generated representation
return P

a backbone clustering algorithm from options such as K-means, K-medoids, and hierarchical clustering. However, it's worth noting that hierarchical clustering has a drawback that it cannot undo previous clustering decisions [107]; in other words, once sub-clusters are merged, this merging cannot be modified, even if it later proves to be suboptimal. Additionally, its computational inefficiency makes it impractical to use on large datasets [93, 94]. K-means and K-medoids are similar approaches, but the time complexity of K-medoids is much higher than that of K-means [150], and this is undesirable for processing large time series. Based on these issues, K-means is chosen for TDC; the details of backbone algorithm comparison are discussed in Section 3.3.2. For applying K-means, the similarity between sequences is measured by Euclidean distance on the generated representation. For the entire workflow of TDC, the pseudo code is shown as Algorithm 1.

3.2.2 Baselines and metrics

For assessing the performance and complexity of TDC, we have selected three well-known methods: K-means, K-medoids and hierarchical clustering which have been discussed in the previous chapter. For hierarchical clustering, there are different linkage criteria to create the clusters; here, we applied hierarchical clustering with average linkage which is widely used in applications. These three methods have been used employing the original time series as input; the objective is to assess the performance and complexity of these baseline approaches with respect to TDC.

To evaluate the performance of clustering algorithms, we use four different criteria. The first one is the running time of each algorithm, which is a straightforward way to compare the execution time and computational cost. Besides the running time, clustering metrics are used to evaluate the quality of the generated clusters:

- Silhouette Value [151]: this is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation), and it ranges between -1 and 1. The higher the silhouette value, the better the separation. A negative value means the clusters are separated in a wrong way.
- Davies–Bouldin Index [152]: this index is defined as the average similarity measure of each cluster with its most similar cluster, and the index ranges from 0 to ∞. A lower Davies–Bouldin Index indicates a better clustering result.
- Calinski-Harabasz Index [153]: this index measures the ratio of the sum of between-clusters dispersion and intra-cluster dispersion for all clusters, and its value ranges from 0 to ∞; the higher the index, the better the clustering is.

3.3 Experiments and results

In this section, experiments are carried out and the related findings are discussed. We perform clustering on two datasets: we first use the larger dataset to determine the backbone algorithm of TDC and evaluate the performance of TDC and baselines with the aforementioned metrics, and then compare the behaviour of TDC and K-medoids in detail. After that, a small dataset is used to visualize the clustering result of TDC and analyze usage pattern of cells in the city of Turin (Italy). To

implement the baselines, we use the following libraries: K-means (Tslearn [154]), K-medoids (PyClustering [155]) and hierarchical clustering (Scikit-learn [156]). All experiments are conducted on a computer with Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz and 64 GB memory.

3.3.1 Dataset

In our experiments, we use two real-world mobile network datasets and one air quality dataset. Both mobile network datasets are collected and processed by Telecom Italia S.p.A which is the largest italian telecommunications services provider; the datasets have been preprocessed to deal with missing values and standardize data. The first dataset consists of 16300 multivariate time series, and the records are collected from 100 cells in LTE network; three frequency layers are controlled by these cells, including 28 2600MHz cells, 43 1800MHz cells and 29 800MHz cells. The length of each sequence equals 1440 samples collected in 15 days. The second dataset monitors the activity of cells over 2 weeks from 09 November 2020 to 22 November 2020. 956 multivariate time series are collected from different cells in Turin, and the length of each time series is 1344 samples; within the second dataset, we have 349 2600MHz cells, 493 1800MHz cells and 114 800MHz cells. In this study, we are interested in two important network parameters: the downlink usage and the average number of connected users. Clustering algorithms are performed based on these two variables. To investigate the generalization of the proposed algorithm, TDC is also applied on a public dataset of air quality [157]; we subsample this dataset to obtain 4251 time series of the concentration of PM2.5, and each sequence has 336 samples (two weeks observations of hourly data).

Notice that all time series of these datasets have been normalized with the standard score normalization, and the normalized time series is calculated by first subtracting the mean value of raw time series and then dividing by the standard deviation.

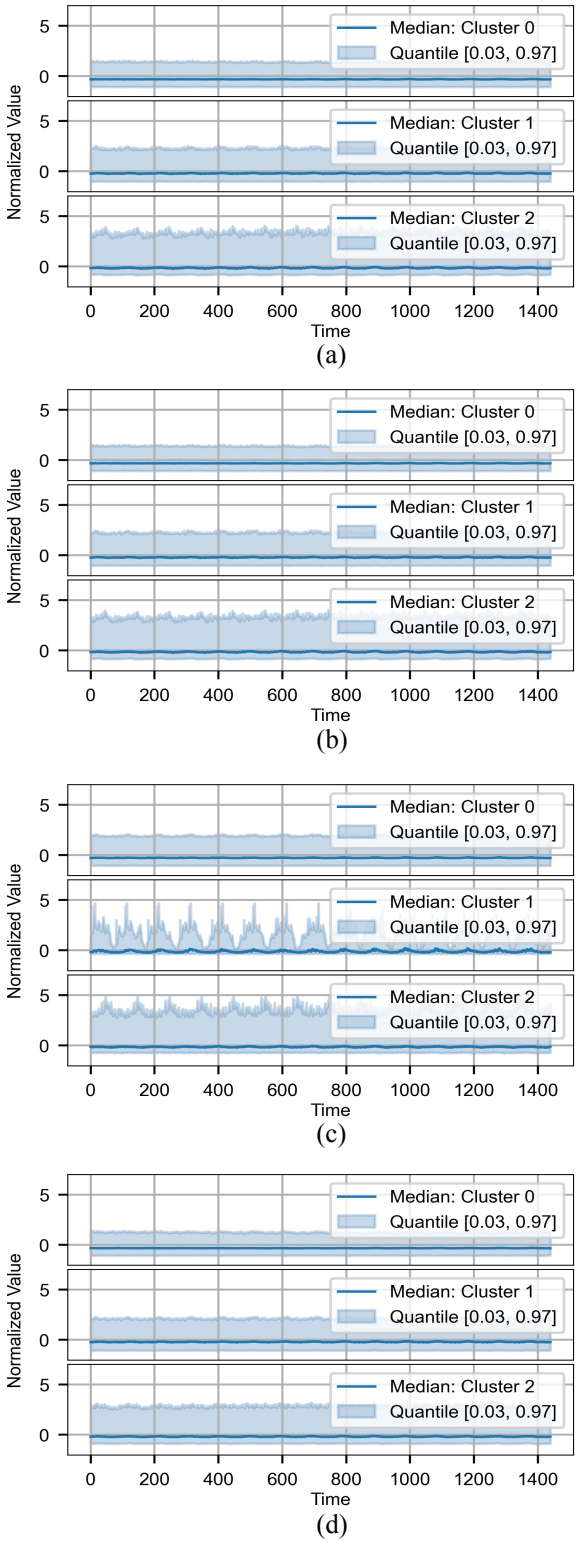


Fig. 3.4 Comparison of backbone algorithms (time series of downlink usage are plotted). From top to bottom: cluster 0, cluster 1 and cluster 2. The blue line is the median value calculated within the cluster, and the shaded region indicates the upper bound 0.97 quantile and the lower bound 0.03 quantile of each cluster. (a) K-means; (b) K-medoids; (c) Hierarchical clustering: Average Linkage; (d) Hierarchical clustering: Ward Linkage.

Backbone Algorithm	Running Time	Silhouette	DB Index	CH Index
K-means	**4.65 mins**	0.435	0.785	**18417.76**
K-medoids	5.94 mins	0.423	0.789	18403.98
Hierarchical: Average Linkage	4.73 mins	**0.591**	**0.543**	4541.66
Hierarchical: Ward Linkage	4.74 mins	0.363	0.882	14519.89

Table 3.1 Comparison of backbone clustering algorithms.

3.3.2 Comparison of backbone algorithms

As mentioned in Section 3.2.1, we evaluate the performance of various approaches to select the backbone clustering algorithm for TDC. The results are reported in Table 3.1. We perform TDC with four backbone approaches: K-means, K-medoids, hierarchical clustering with average linkage and hierarchical clustering with ward linkage. From Table 3.1, hierarchical clustering with ward linkage is the worst one among these approaches, while the average linkage version has the highest silhouette score and DB index, but its CH index is much lower than the others. Compared to K-means, K-medoids have similar performance of clustering metrics; however, the running time of K-medoids is much longer than K-means which is undesirable. According to these metrics, K-means and hierarchical clustering are good candidates as the backbone method. To compare these algorithms in detail, Figure 3.4 illustrates the behaviour of clusters generated by different algorithms. From the figures, K-means, K-medoids and hierarchical clustering with ward linkage generate similar clusters; these approaches split time series into different groups based on their temporal dynamics. If we perform hierarchical clustering with average linkage, we cannot identify the temporal dynamics level from Figure 3.4(c); it is difficult to compare these clusters when their behaviour differs a lot, especially when one of them has a sinusoidal shape. Based on metrics and cluster behaviour, K-means is the best one and it is selected for TDC.

3.3.3 Evaluation of TDC

Comparison of clustering approaches

In this subsection, all experiments have been done on the first mobile network dataset. We first apply TDC to generate three clusters. For studying the behaviour of this

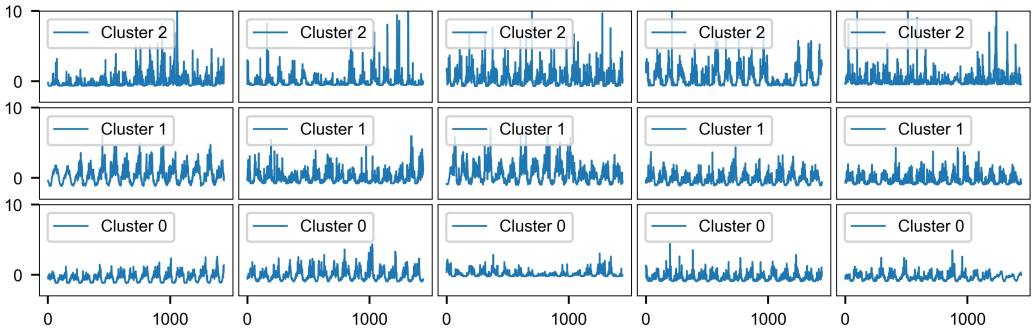


Fig. 3.5 Snapshot of clusters generated by TDC (time series of downlink usage are plotted), where x axis represents time and y axis represents the normalized score. The first row refers to cluster 2, the second row refers to cluster 1 and the last row refers to cluster 0.

algorithm, Figure 3.5 provides a visual representation of clusters. It can be seen that cluster 2 is the cluster whose temporal dynamics is the highest one. If we move from cluster 2 to cluster 0, the oscillation of time series becomes much weaker, and a decreasing trend of temporal dynamics is observed. According to the plots, cluster 2 is more important than the others for mobile networks, as the sequences belonging to this cluster suffer a heavier mobile demand because of the occurrence of spikes. In this case, it is very likely that spikes represent specific events which could be caused by user movement or social events.

To have a comprehensive view of our algorithm, we compare the behaviour of clusters generated by TDC, K-medoids clustering and K-means clustering, and the result is shown in Figure 3.6 and Figure 3.7. From Figure 3.6, the median value and variation range of each cluster are plotted to provide a global perspective of cluster behaviour. Based on the figures, TDC generates clusters based on the variability of time series. Among these clusters, cluster 2 has the highest, and cluster 0 exhibits the weakest oscillation. Comparing with TDC, the behaviour of K-medoids is very different. K-medoids generates very similar clusters, as the algorithm measures the similarity between original time series with Euclidean distance, which results in the waveform-oriented grouping. Even though we can observe the difference between resulting clusters, their patterns are quite similar, and the grouping rule is mainly derived by the horizontal shift of the waveform. Comparing the behaviour of clusters created by TDC and K-medoids, TDC performs clustering with “high dimensional" features which are only available after mining knowledge from the original input, and the steps of feature extraction provide the clustering engine a different way to

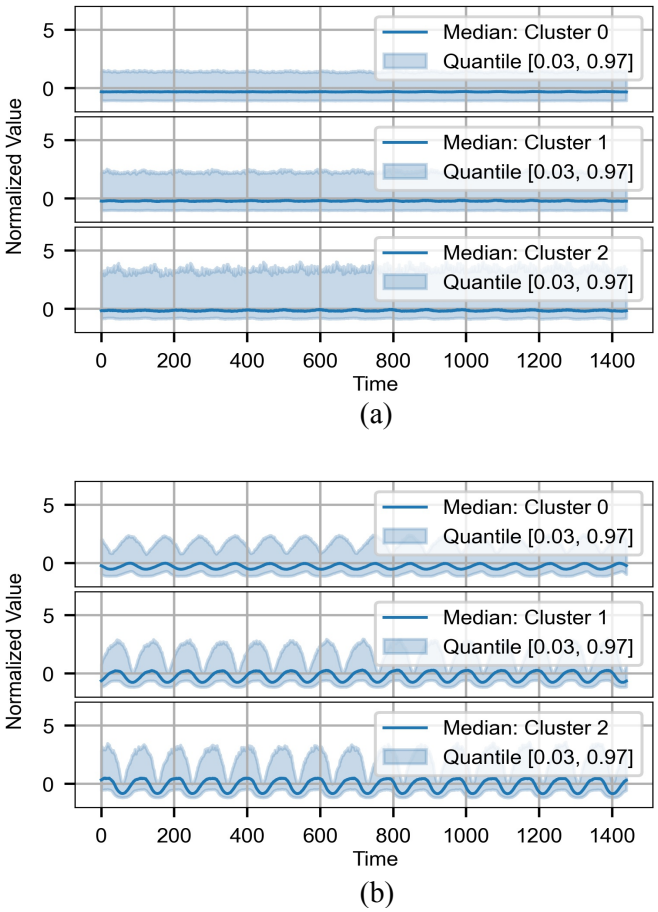


Fig. 3.6 Comparison of TDC and K-medoids clustering (time series of downlink usage are plotted). From top to bottom: cluster 0, cluster 1 and cluster 2. The blue line is the median value calculated within the cluster, and the shaded region indicates the upper bound 0.97 quantile and the lower bound 0.03 quantile of each cluster. (a) TDC; (b) K-medoids Clustering.

understand the same input, enabling a more creative separation of clusters. As can be seen in Figure 3.7, the clusters generated by K-means are very similar to the clusters of K-medoids, as these two approaches cluster sequences following the similar rule. To evaluate the performance of our algorithm, we compare the performance of TDC and baselines based on the metrics mentioned above, and the result is reported in Table 3.2.

From Table 3.2, TDC outperforms other algorithms on two metrics. The results of K-means and K-medoids are similar, and their performance is quite poor. Hierarchical clustering has the highest silhouette score among these algorithms, but its

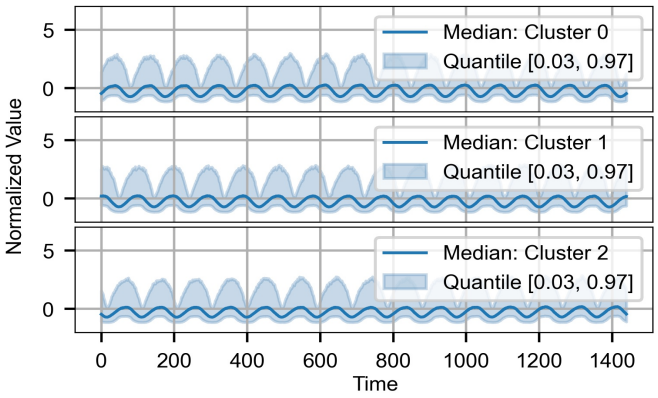


Fig. 3.7 The behaviour of the clusters created by K-means (time series of downlink usage are plotted).

Algorithm	Silhouette	DB Index	CH Index
K-means	0.137	1.949	2652.16
K-medoids	0.125	2.089	2223.17
Hierarchical	**0.616**	1.769	84.21
TDC	0.435	**0.785**	**18417.76**

Table 3.2 Comparison of clustering algorithms.

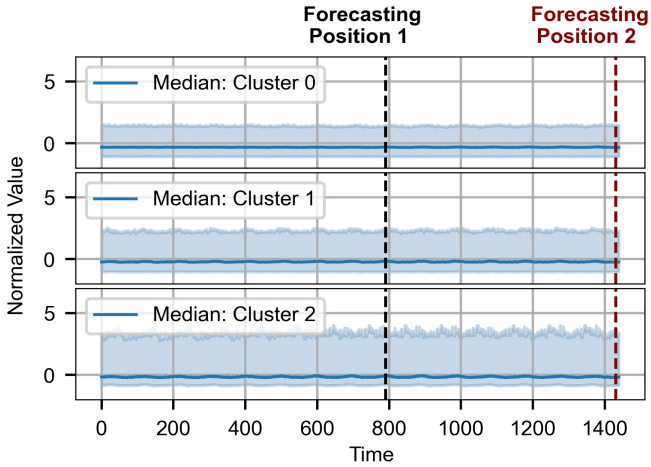
CH index is much lower than the others. Overall, TDC is the best one among them. According to the results, the use of the proposed feature extractor significantly boosts the clustering performance, and the clustering engine can generate clusters in a better separated way in the latent space. Based on the previous results, a well-designed feature extractor can be more important than a clustering algorithm in the sense of improving clustering performance or customizing the behaviour of clusters. Besides the metrics related to the separation of clusters, running time is another key issue of applying clustering algorithms on large datasets, as some algorithms are essentially not capable of dealing effectively with large time series. To study the behaviour of these algorithms, the running time of baselines and TDC are shown in Table 3.3. There are three terms in the table: precomputation time is the time used to perform precomputation, such as the time of calculating pairwise distance matrix (K-medoids and hierarchical clustering) or generating representation of time series (TDC). According to this table, the computational cost of creating the distance matrix is very expensive when the dataset is large. The calculation takes approximately 7.8 hours even with Euclidean distance, which means DTW is impractical to use

Running Time	**Precomputation**	**Clustering**	**Total**
`K-means`	—	0.30 minutes	**0.30 minutes**
`K-medoids`	7.79 hours	2.13 minutes	7.82 hours
`Hierarchical`	7.79 hours	0.12 minutes	7.79 hours
`TDC`	**4.50 minutes**	**0.01 minutes**	4.51 minutes

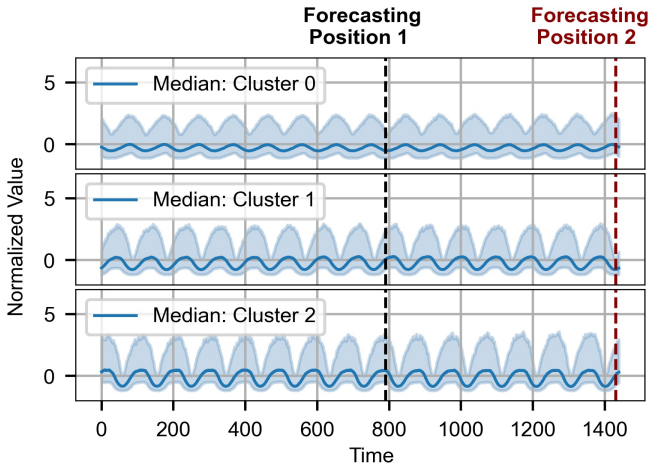
Table 3.3 Running time of clustering algorithms.

considering its complexity is quadratic. Among these algorithms, K-means clustering has the shortest running time which is 0.3 minutes. For K-means, there is no need to calculate the pairwise distance between all objects, and this reduces the computational cost significantly. Comparing K-medoids and hierarchical clustering, hierarchical clustering is computationally cheaper due to the lower time complexity $O(m^2)$ [94]. K-medoids is much more expensive, as the complexity is $O(m^2z)$ for BUILD phase and $O(i(m-z)^2z)$ for SWAP phase, where m is the number of objects, z is the number of clusters and i is the number of needed iterations [150, 107]. The result proves that the similarity measurement is more important than the clustering itself if we want to reduce the overall computational cost, and that is one reason why TDC generates a new representation before applying K-means. For TDC, the creation of representation takes 4.50 minutes, and its K-means phase is 30x faster (0.01 minutes) than the basic one (0.30 minutes) with the new representation. In Algorithm 1, the time complexity of creating representation is $O(mqg^2)$, where q is the number of variables and g is the length of each sequence. The calculation of quantile is the most computationally consuming operation because it requires sorting the sequence first, the time complexity of sorting is sequence-length dependent which usually ranges from $O(g\log(g))$ to $O(g^2)$ depending on sorting algorithms. The total running time of TDC is slower than K-means but 100x faster than hierarchical clustering and K-medoids. Even though K-means is a fast algorithm, it cannot generate well-separated clusters and compare the variability properly (Figure 3.7), which makes it unsuitable for analyzing the cell importance. Notice that the frequency at which network KPIs are acquired is approximately 15 minutes or more in the system, which means the difference between 0.30 minutes (K-means) and 4.51 minutes running time (TDC) is quite small, and both of them are fast enough if we want to perform real-time analysis, obtaining the results before the next KPI is acquired. Compared to K-means, TDC has an additional pre-processing cost as it requires to generate representation before running clustering; however, the clusters

created by TDC are much better than the ones of K-means, and K-means cannot identify the variability at all; in this case, the minor additional cost of pre-processing is acceptable. Considering both the quality of clusters and total running time, our algorithm is the best approach among them.



(a)



(b)

Fig. 3.8 Two predefined forecasting positions to evaluate the forecasting difficulty of clusters. (a) TDC; (b) K-medoids Clustering.

Analysis of forecasting difficulty

For TDC, besides the aforementioned advantages, the algorithm can also identify the forecasting difficulty of time series which provides supplemental information for

improving and evaluating prediction models. If we want to make accurate time series predictions, how to make a forecasting model better learning the temporal dynamics of time series is a critical problem. Usually the time series with strong temporal dynamics are more difficult to predict compared to the others, and the prediction performance can be improved a lot if the architecture of the model is optimized for predicting these sequences. However, due to the fact that we do not know which time series are more difficult to predict, it is hard to improve the forecasting model and study the forecasting behaviour in detail. From this point of view, TDC is a useful tool which provides us the knowledge of the forecasting difficulty of time series. By considering clusters are created from the variability of time series, each cluster can be seen as a collection of time series which has different amounts of dynamics. For time series forecasting, it is very difficult to predict a time series when it is highly stochastic and strongly oscillating. In this case, the evaluation of forecasting difficulty is consistent with the underlying grouping rule of TDC. To verify this, we perform experiments aimed at evaluating the prediction difficulty of different clusters. In Figure 3.8, we define two forecasting positions at which a purposely trained neural network is tested for prediction accuracy. At forecasting position 1, the goal is to use the preceding 796 historical measurements to predict the values of the next 4 steps. At forecasting position 2, the goal is to use the preceding 1436 historical measurements to predict the values of the last 4 steps. The reason why we define different forecasting positions is to prevent forecasting errors from being biased by local waveform. For example, If we observe Figure 3.8(b), we can find that the oscillating strength of clusters changes over time; for the peak parts of the signal, the median value is not close to the 0.97 quantile, and the distance between them decreases a lot at the valley parts of the signal. In this case, the forecasting difficulty of each cluster is not consistent over time considering the temporal dynamics change periodically, and it is essential to define multiple forecasting positions to evaluate the performance of the predictor.

Following this idea, we train a MLP individually on each cluster generated by TDC and K-medoids. MLP is a feedforward neural network which is composed of an input layer, hidden layers and an output layer, and it is widely used in research because it can approximate solutions for complex problems. In our experiments, MLP consists of three fully connected layers and ReLU is used as the activation function; MLP is trained to predict the downlink usage and the average number of connected users in next hour by using their historical measurements. For each cluster,

we shuffle its time series and use 80% of them as train set, 10% of them as validation set and the other 10% of them as test set. We first train MLP on the train set, and the training is stopped if the Mean Absolute Error (MAE) cannot be decreased on validation set. The hyperparameters of MLP are tuned based on its performance evaluated on validation set, then with these hyperparameters, MLP is retrained on the merged original train set and validation set. The performance of MLP is evaluated on the corresponding test set and the results are reported in Table 3.4 and Table 3.5.

	Cluster 0	**Cluster 1**	**Cluster 2**
`Forecasting Position 1`	$\mathbf{0.199 \pm 0.034}$	0.362 ± 0.051	0.376 ± 0.040
`Forecasting Position 2`	0.289 ± 0.044	$\mathbf{0.169 \pm 0.018}$	0.390 ± 0.037

Table 3.4 MAE of MLP evaluated on the clusters created by K-medoids.

	Cluster 0	**Cluster 1**	**Cluster 2**
`Forecasting Position 1`	$\mathbf{0.214 \pm 0.022}$	0.329 ± 0.047	0.444 ± 0.055
`Forecasting Position 2`	$\mathbf{0.220 \pm 0.030}$	0.312 ± 0.056	0.438 ± 0.083

Table 3.5 MAE of MLP evaluated on the clusters created by TDC.

The two tables show MAE calculated on the test set of each cluster, and the performance of predictors differs a lot on clusters generated by different algorithms. For the clusters generated by K-medoids (Table 3.4), at forecasting position 1, the predictor gets the best performance on cluster 0, and the easiest dataset changes from cluster 0 to cluster 1 if we switch to the second forecasting position. When we go back to Figure 3.8(b), it is easy to notice that cluster 0 has the lowest variation at forecasting position 1, and for forecasting position 2, the lowest variation can be found on cluster 1. This verifies our assumption which is mentioned in the previous discussion: the difficulty of forecasting could change with the forecasting position of the waveform. In this case, the clusters generated by K-medoids cannot be used to describe the difficulty of time series because the performance of the predictor evaluated on each cluster is not consistent over time. Compared to K-medoids, the behaviour of clusters created by TDC is consistent; irrespective of which forecasting position is used to evaluate, cluster 0 is always the easiest one which relates to weak temporal dynamics, and we can observe an increasing trend of MAE with the increment of temporal dynamics level.

	Cluster 0	**Cluster 1**	**Cluster 2**
`50% of Training Data`	0.232 ± 0.031	0.346 ± 0.052	0.446 ± 0.102
`75% of Training Data`	0.231 ± 0.032	0.327 ± 0.040	0.449 ± 0.109
`100% of Training Data`	$\mathbf{0.220 \pm 0.030}$	$\mathbf{0.312 \pm 0.056}$	$\mathbf{0.438 \pm 0.083}$

Table 3.6 MAE of MLP evaluated on the clusters created by TDC (forecasting position 2).

The forecasting difficulty can be seen as an inherent characteristic of time series, and it is difficult to improve the prediction of time series by simply changing the size of training data, especially when the forecasting difficulty is high. To verify this point, we change the size of the datasets used to train MLP, and the results are shown in Table 3.6. According to the results, we can get minor improvement of the prediction if we use more data to train MLP; however, the improvement is too small to compensate the additional computational cost. The improvement is slightly larger for the relatively easier clusters (cluster 0 and cluster 1), while the improvement for the difficult cluster (cluster 2) is essentially negligible. From this point of view, the potential improvement of time series prediction is related to the corresponding forecasting difficulty, and the approach of improving prediction performance by using more training data is neither efficient nor computational economic.

Investigation of generalization

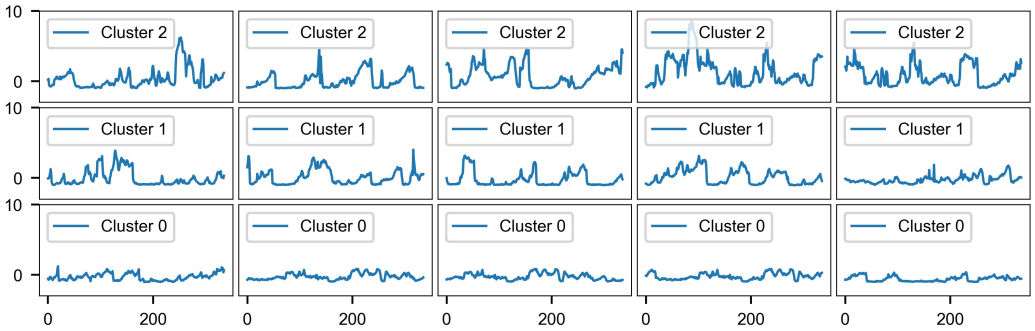


Fig. 3.9 Snapshot of clusters generated by TDC (PM2.5 dataset), where x axis represents time and y axis represents the normalized score. The first row refers to cluster 2, the second row refers to cluster 1 and the last row refers to cluster 0.

Algorithm	Running Time	Silhouette	DB Index	CH Index
K-means	**0.03 mins**	0.253	2.695	386.87
K-medoids	1.93 mins	0.122	3.936	160.41
Hierarchical	1.82 mins	0.449	1.525	160.41
TDC	**0.03 mins**	**0.631**	**0.481**	**12775.32**

Table 3.7 Comparison of clustering algorithms (PM2.5 dataset).

To study the generalization of TDC, we also apply TDC on another dataset which is composed of time series of PM2.5. Figure 3.9 illustrates the clustering results, and we can observe that TDC creates different clusters based on the variability of PM2.5 time series and the behaviour of the clusters is similar to the previous result of the mobile network dataset. The result demonstrates that TDC exhibits consistent behaviour and it can also be applied with the data which is not related to mobile networks. Table 3.7 compares the clustering performance of TDC and baselines, and the result shows that TDC outperforms all the other methods. On this dataset, TDC is as fast as K-means by considering its precomputation time decreases a lot as the time series is much shorter (the time series length is 1344 for mobile data and 336 for PM2.5 data).

3.3.4 Analysis of cell behaviour in urban environment

In the previous section we discussed the characteristics of TDC and evaluated its performance using different metrics. In this section, we focus on the real usage case of this algorithm of analyzing the cell behaviour in intricate urban environments. Due to the complex usage patterns of mobile users, the distribution of mobile traffic is not homogeneous in a telecommunication network, whereby some cells carry more traffic and serve more users than others. For better allocating network resources, it is crucial to understand the importance of each cell, so that suitable resources can be allocated. In this section, we analyze the traffic pattern of cells located in Turin.

We first study the overall behaviour of cells with two two-week measurements, and then we explore the time-specific behaviour of cells by using records collected in different time slots. Figure 3.10 visualizes the overall importance of cells in Turin; the cells belonging to cluster 2 are the most important ones, they exhibit strong temporal dynamics which means they suffer heavily changing mobile demand.

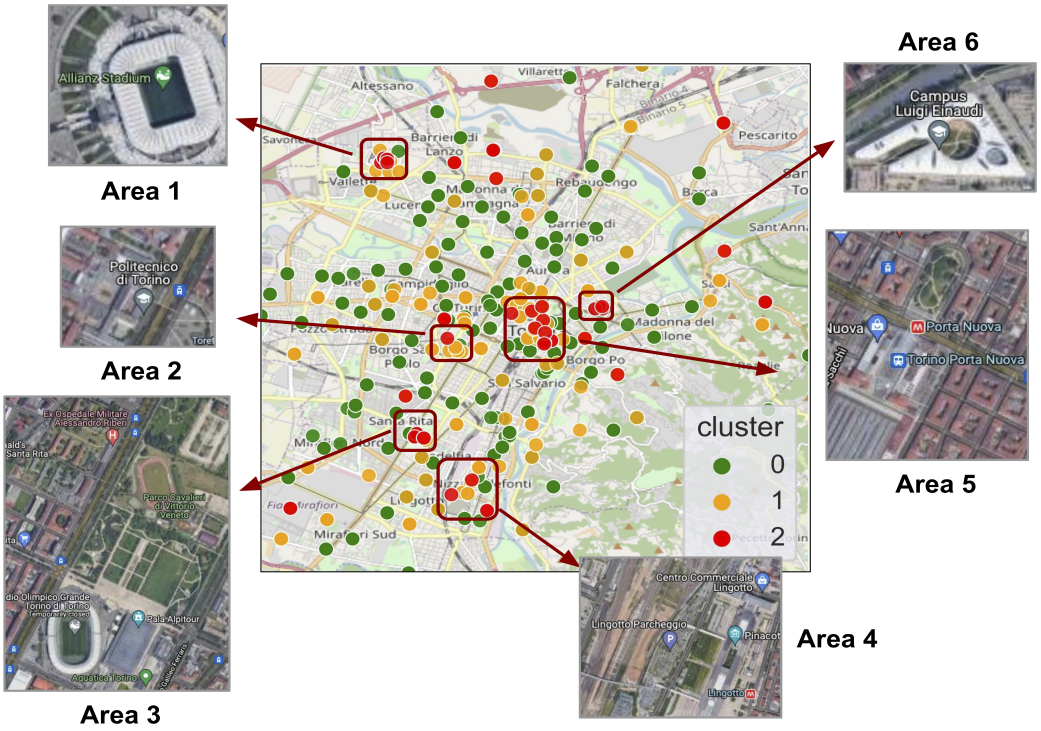


Fig. 3.10 Visualization of the geographical distribution of clusters in urban areas, each dot represents one deploying location of cell. Among the clusters, cluster 2 has the strongest temporal dynamics while cluster 0 has the weakest one.

On the map, most cells are grouped into cluster 0 (green dots) which is the least important cluster from a global view, these cells are mostly distributed among residential areas and suburbs. For moderately important cells (yellow dots) and the most important cells (red dots), the size of these clusters is much smaller than that of the green one. Observing the geolocation of cells, it can be seen that some areas are more important because the neighbouring cells have higher temporal dynamics. To understand the hidden patterns, we explore the land usage of 6 different areas in detail. Employing Google Maps we find that the importance of cells mainly relates to human activities and functionalities of the corresponding area, especially for the regions whose functionalities are related to sport, work/study, commute and leisure. For example, Area 1, Area 3 and Area 4 play key roles in sport: Area 1 is the stadium of the Juventus football club, Area 3 contains the Olympic stadium and one large park of Turin which is a popular place for jogging. In Area 4 there is another large stadium which was used for figure skating and short track speed skating events at the 2006 Winter Olympics. Among these areas, Area 4 and Area 5 are comprehensive

areas, and both of them have complicated functionalities. Area 5 is the city centre of Turin, which has two train stations, plenty of restaurants, offices and historical buildings. Another train station can be found in Area 4, which also has a large mall. Area 2 and Area 6 only have the functionality of education, and include the main campus of the Polytechnic of Turin and University of Turin. Based on this verification, it is obvious that the importance of cells is highly related to human activities.

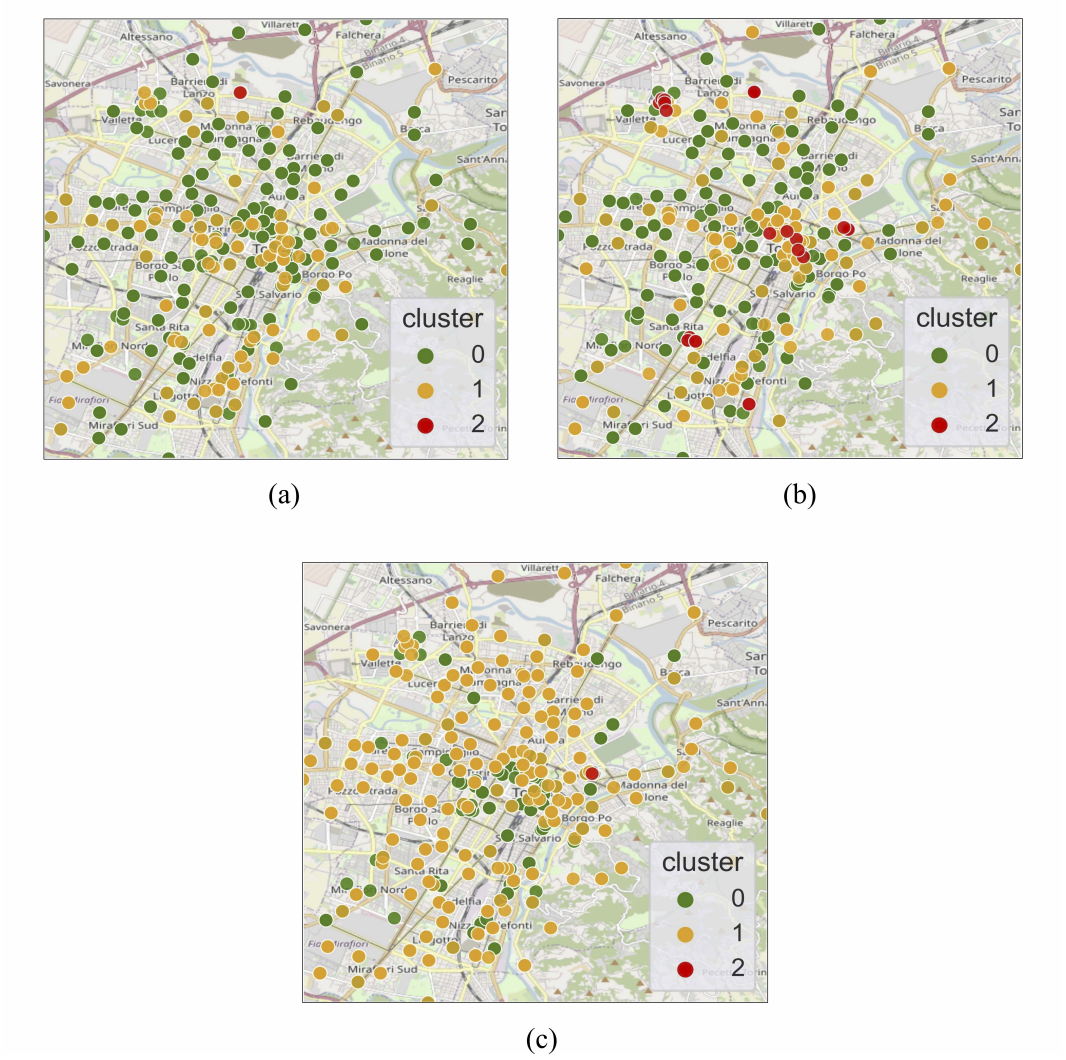


Fig. 3.11 Visualization of the geographical distribution of clusters in different time slots. (a) Morning (from 6:00 to 9:30); (b) Afternoon (from 13:30 to 17:30); (c) Night (from 22:00 to 24:00).

Human activities are time-dependent, and this characteristic makes the usage pattern evolve over time, which results in the change of cell importance. To investigate this aspect, we perform clustering for three time slots: morning (commuting time), afternoon (working time) and night (entertainment time). From Figure 3.11, it can be seen that the cell importance differs a lot in different time slots. In the morning, most of the important cells are near to train stations, subway stations and the main roads of Turin (e.g., the roads along East-West and South-West directions). The afternoon is usually seen as working time, and the pattern shown in Figure 3.11(b) is very similar to Figure 3.10; city centre and stadiums are highlighted in this time slot. During the night, most of the cells are grouped into cluster 1 and cluster 0. Compared with the afternoon, cells distribute in a more homogeneous way. Basically the moderately important cells (yellow dots) include transport facilities and residential areas.

3.4 Conclusions

In this chapter we presented TDC, a novel time series clustering algorithm for analyzing the cell behaviour in city-wide mobile networks. This algorithm measures the variation between adjacent elements by creating first-order differences sequences, and a new representation is generated by summarizing the distribution of differences sequences to represent the temporal dynamics level of time series. The created representation is employed by K-means to split time series into different groups based on their variability. We evaluated the performance of our algorithm on real-world datasets of mobile networks, the results showed that our algorithm is the best one among baselines. Through TDC, we can generate clusters based on temporal dynamics level in a computationally efficient way, which is very beneficial for performing clustering analysis on large datasets of mobile networks. Our algorithm can also identify the forecasting difficulty of time series, providing supplemental information for improving and evaluating prediction models.

To study the cell behaviour in urban environments, we applied TDC to discover the usage pattern of mobile network cells in Turin, Italy. Through detailed analysis of several relevant areas, we found that the importance of cells relates to the land usage of that region. The areas whose functionality is closely tied to sport, work/study, commute and leisure play a key role in urban environments, the cells which are deployed in these regions are more important in mobile networks, and the behaviour

of these cells evolves over time following the time-dependent nature of human activities.

In this chapter, we proved that our algorithm is effective for addressing real-world problems, and provided a deeper understanding of traffic usage patterns in intricate urban environments, which is very valuable for mobile operators.

Chapter 4

Prediction of mobile traffic peaks and imbalance

In the last chapter, we presented how to extract knowledge and study cell behaviour using TDC, which provides mobile operators with a deeper understanding of mobile usage patterns. Starting from this chapter, we will present the solutions developed for improving mobile traffic forecasting - a critical topic in automated mobile network maintenance. This chapter studies the subjects of predicting mobile traffic peaks and peak-related events, which are worthwhile studying but have not been investigated well due to their difficulties. To perform predictive maintenance of mobile networks, accurate mobile traffic prediction is crucial for resource allocation. In adjacent cells, mobile traffic may occasionally concentrate in a single cell, resulting in traffic imbalance. Detecting this imbalance is often closely linked to predicting traffic peaks, a task fraught with difficulty as many peaks occur suddenly and without apparent cause. If we can predict the mobile traffic peaks with higher accuracy, it is more able to address the network issues raised by unexpected events and improve the QoS. In this chapter, the material appeared in my publications [158, 159].

4.1 An essential trade-off of forecasting

Over the past decade, there has been a significant surge in global mobile traffic, driven by a substantial increase in mobile devices and their applications, coupled with advancements in modern cellular networks; as video streaming applications

continue to gain popularity, mobile video traffic now surpasses half of all mobile data traffic, placing greater demands on base stations to sustain higher mobile usage than ever before. Compared to other mobile services, streaming services have stricter QoS requirements; improving QoS and maintaining the stability of mobile networks are becoming critical concerns for mobile network operators.

Among the network performance indicators of LTE networks, the QoS experienced by mobile users is closely linked to downlink throughput, a metric whose value is influenced by numerous factors. In practice, the network operator is interested in improving not only the average user throughput of the networks but also the worst 5% user throughput [160, 161]. In mobile networks, each eNodeB consists of multiple cells, which are connected targets of mobile devices. For a set of cells, it is not rare that the mobile traffic is distributed among them in an imbalanced way. In such scenarios, most of the traffic is concentrated on a single cell, resulting in congestion and a decreased throughput for mobile users allocated to that specific cell; from this perspective, achieving a balanced traffic distribution among cells is preferable, as it would improve the QoS for the congested cell. In practice, a group of adjacent cells within a specific geographical area can be seen as a small cluster, and it is feasible to redistribute traffic from a congested cell to its adjacent ones if the traffic distribution among them is imbalanced.

To configure cells and manage the mobile traffic, two methods which have been widely used in literature are Coverage and Capacity Optimization (CCO) and Mobility load balancing (MLB) [162, 163]. CCO [164, 165] finds the proper radio parameters to satisfy the requirements of both the coverage and the capacity, whereas MLB [166] modifies the cell-specific offset between neighbouring cells adaptively depending on their load difference; if the load difference exceeds the presetting threshold, then the algorithm will be triggered to control the traffic handover between different cells. With these techniques, a mobile network operator can adjust the handover offset or switch between predefined configurations to encourage traffic redistribution. However, regardless of the method used to control traffic handover, there always remains a challenge related to the decision-making. Typically, there is a delay in obtaining the latest network measurements, making it risky to rely on these delayed observations to reconfigure the network; updated configurations based on outdated data may not effectively adapt to the actual traffic patterns, especially during traffic peaks that significantly impact QoS. From this perspective, using the predictions of mobile traffic could be more beneficial than using the latest

observations. Moreover, predictions can also enhance the stability of the system which is crucial for network operators. If we can obtain accurate mobile traffic predictions, the resource allocation can be done more efficiently. For load balancing, mobile traffic predictions enable intelligent load balancing across different cells. By distributing the traffic more evenly, operators can prevent network hotspots and ensure that resources are utilized efficiently. The predictions can also be used to achieve dynamic resource provisioning, where mobile networks can dynamically allocate resources based on real-time traffic predictions. For instance, during a large-scale event or a sudden increase in mobile traffic, additional resources can be provisioned on-the-fly by modifying the cell configuration to meet the increased capacity requirements; alternatively, cells that are handling very little traffic can be switched off to save energy.

Even though the accurate prediction of traffic peak is very important, most of the related research focuses on improving the overall prediction performance instead of the peak forecasting performance [116, 167, 168]. DL-based approaches are seen as promising methods for forecasting as they have been proven to be very effective at learning complex non-linear patterns; many DL models have been used to predict time series in recent years [60]. On the other hand, while these models are indeed powerful, they are unable to handle the trade-off between the average accuracy and the accuracy of peak predictions. Mobile traffic time series typically exhibit a generally stationary behaviour interrupted by sudden strong peaks; many time series forecasting methods are very good at predicting the stationary parts but provide inaccurate results in the prediction of peaks. The reason lies in the fact that forecasting models are often trained to minimize an average loss on the prediction error. Since peaks occur rarely, their prediction tends to be neglected because their effect on the average loss is quite small, whereas the loss is dominated by errors in the stationary parts of the mobile traffic time series; this leads to models typically making conservative predictions most of the time. If the model predicts the future steps aggressively, this would improve the prediction of future peaks but also make the overall performance worse as aggressive predictions are always risky. Eventually, there is always a trade-off between overall loss and peak prediction, which has to be handled according to application requirements. For mobile operators, it is acceptable that the model makes better peak predictions even though the errors on the non-peak parts are slightly higher, as the peaks are the most valuable parts of mobile traffic.

To tackle these challenges and improve the traffic management of mobile networks; in this chapter, we propose a new method to detect and adjust the potential traffic imbalance within a cluster of cells. Our solution is composed of two different phases: the mobile traffic forecasting phase and the traffic imbalance detection phase. In the first phase, two DL models can be selected to predict future traffic, namely Mixture of Quantiles (MoQ) and Flexible Multilayer Perceptron (FMLP). MoQ is built based on the MoE framework supporting the flexible blending of different forecasting styles, where aggressive and conservative forecasting are adaptively aggregated based on the recent temporal dynamics of the time series. Through the cooperation between experts with diverse characteristics, this model is capable of making better peak predictions while maintaining excellent overall performance and interpretability. Since MoQ is a computationally expensive model, we also propose the lightweight model - FMLP. FMLP consists of a predictor and a data mapping module, which selects important samples and performs conditional magnitude scaling on the time series; this allows the model to perform forecasting in a flexible way. The behaviour of this model can be easily controlled by changing a pre-defined scaling factor, resulting in different levels of aggressiveness in the predictions. In the traffic imbalance detection phase, the predictions of cells are used by a detection algorithm to determine if there will be a potential traffic imbalance; in this chapter, two predictive detection approaches are discussed: the single-model approach and the multi-model approach. Compared to the naive approach which uses the recent observations instead of the predictions, both of the predictive solutions achieve better results using real-world mobile traffic data. Specifically, the main contributions of this chapter are summarized as follows:

- A predictive approach to detect the potential mobile traffic imbalance among cells combining a mobile traffic predictor with an imbalance detection algorithm.
- A novel mobile traffic forecasting model called MoQ features a flexible blending of conservative and aggressive predictions based on recent observations.
- A novel lightweight mobile traffic forecasting model called FMLP focuses on learning important patterns; The model behaviour can be tuned from conservative to aggressive prediction.

- A large-scale analysis is performed to study the traffic behaviour within the clusters, and extensive experiments are carried out on real-world mobile traffic datasets, proving the advantages provided by the proposed methods.

4.2 Background and problem formulation

4.2.1 Mobile traffic management and network configuration

For topology planning and mobile traffic management, MLB and CCO are two popular approaches. CCO solutions focus on adjusting the antenna tilt to optimize the capacity and coverage, thereby impacting network performance and traffic assignment [169–171]; how to adjust the antenna tilt efficiently is the major concern in this field [165, 172, 173]. Unlike CCO, MLB does not involve configuring antennas; instead, it modifies the handover region between adjacent cells, resulting in a reduced call drop rate and improved QoS [166, 174–176]; few papers have addressed the issue of detecting potential traffic load imbalances. In this chapter, we introduce a traffic prediction-based approach to determine whether there is a need to redistribute load among neighboring cells in the near future.

4.2.2 Mixture of experts

MoE model was first proposed in [177], where the authors introduce an ensemble style framework which consists of many sub-networks (experts) and a gating network (manager). In a MoE model, each expert is expected to learn different knowledge from the training data, and their outputs are blended by the manager to leverage the benefits of diverse patterns. This concept has garnered significant attention in both the DL and ML domains. [178, 179] build MoE-style models on top of ML algorithms such as support vector machines. [180] proposes deep MoE models for speech enhancement. In natural language processing, [181] introduces a very large sparsely-gated MoE and obtains significantly better results for machine translation tasks. MoE has also proved its effectiveness for computer vision tasks such as image classification and person re-identification [182, 183]. Another solution is to build a layer-level MoE as in [184]; this work creates a DL model composed of multiple MoE layers where each MoE layer is an individual MoE network, and the layers

are stacked together. In this chapter, a model-level MoE is built, introducing a cooperation mechanism enabling MoQ to make good peak predictions.

4.2.3 Quantile loss & quantile regression

For a real-valued random variable Y, denote as $F(y)$ its cumulative distribution function. Given a quantile index $\tau \in (0,1)$, the τ-quantile $q^{(\tau)}$ is defined as $q^{(\tau)} = F^{-1}(\tau)$. For example, the 0.5-quantile $q^{(0.5)}$ is the median. For many applications, risk and uncertainty have to be quantified to make optimal decisions. The need for modelling distribution leads to the use of Quantile Regression [185]. The purpose of quantile regression is to predict the conditional τ-quantile $\hat{y}^{(\tau)}_{t+k}$ given the past observations $y_{:t}$ and a predefined quantile index $\tau \in (0,1)$. Compared to probabilistic forecasting, quantile regression is more robust because it makes no assumption on the data distribution [186]. In order to predict the quantile, the model has to be trained to minimize the total Quantile Loss (also called pinball loss):

$$QL_\tau(y, \hat{y}^{(\tau)}) = \tau(y - \hat{y}^{(\tau)})_+ + (1-\tau)(\hat{y}^{(\tau)} - y)_+, \tag{4.1}$$

where $(\cdot)_+ = \max(0, \cdot)$, and $\hat{y}^{(\tau)}$ is modelled by function $g_\tau(\cdot)$ which can be approximated with a DL model. Many works have been done in the quantile regression field. [187] proposes a model to make quantile forecasting of the load of smart grids, and a new approach is proposed to determine the prediction intervals. [188] combines quantile regression and multitask learning, and extends the application to spatiotemporal problems. In our model design, instead of using quantile predictions to define a prediction interval, quantile predictions are used in an ensemble way, where each expert is trained by minimizing quantile loss with different quantile index τ, resulting in different levels of aggressiveness of forecasting.

4.2.4 Problem formulation: forecasting

In this chapter, we perform short-term time series forecasting using a model trained on historical observations, and the predictions are further used to determine if a traffic imbalance event will occur in the near future. Assuming we want to predict the future values of univariate time series with w forecast horizons, time series is represented as a vector $\mathbf{x}_{1:t} = [x_1, x_2, \ldots, x_t] \in R^t$ which consists of past observations

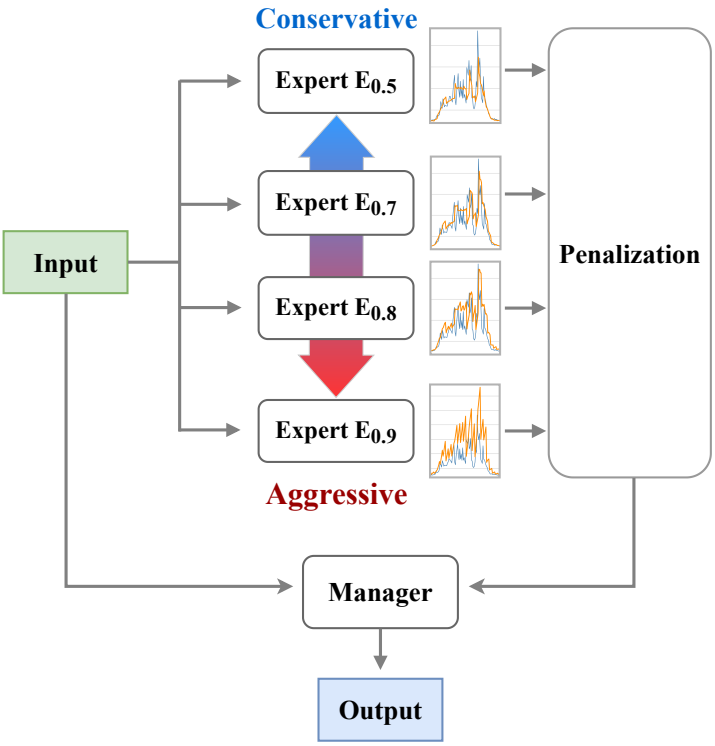


Fig. 4.1 Architecture of Mixture of Quantiles: expert E_q is trained to minimize a quantile loss having a quantile index equal to q. The orange line represents forecasting and the blue line represents the ground truth. MoQ combines the prediction of different experts to yield a more accurate prediction, especially around peaks of the time series.

of the target mobile network KPI, where x_i is the record collected at time step i and t is the length of sequence; the problem can be formulated as:

$$\hat{\mathbf{x}}_{t+1:t+w} = f(\mathbf{x}_{1:t}), \tag{4.2}$$

where $\hat{\mathbf{x}}_{t+1:t+w} \in R^w$ is the vector of predictions from time $t+1$ up to time $t+w$, and $f(\cdot)$ is a selected predictor.

4.3 Mobile traffic predictor: MoQ

4.3.1 MoQ architecture

The architecture of MoQ is shown in Figure 4.1, and it consists of three components:

- several *Experts* which are trained to forecast with different levels of aggressiveness; the outputs of experts are then fed into the manager, which generates the final prediction;
- a *Penalization* applied to prevent the overuse of a specific expert and to promote cooperation among experts;
- a *Manager* aggregating predictions from different experts to output the final forecast.

The idea behind MoQ is to fuse various prediction styles, as illustrated in Figure 4.1. In MoQ, the input is first fed into the experts pool. Through training with different objective functions, these experts have diverse forecasting styles: some of them are going to make aggressive predictions, while others adopt a more conservative style. The manager analyzes the recent temporal behaviour of the input series and fuses these predictions based on the softmax score; this enables the model to automatically adjust between conservative and aggressive styles as needed. Further details of MoQ are discussed in the following sections.

Experts with various forecasting styles

Experts are the key components in MoQ, as their behaviour strongly affects the model performance. In this work, LSTNet [84] is used as the backbone network for each expert. In the MoE framework, the experts are expected to learn different patterns from the same input data. If the characteristics of experts differ a lot, each expert specializes in extracting different knowledge patterns, which provides benefits when the manager aggregates these pieces of information. However, the process raises several issues. The first issue is related to the diversity of experts, as there is no guarantee that each expert will learn different things from the same input. If experts behave similarly, there is little benefit in employing an MoE model. Another issue is about expert assignment; indeed, if we train the MoE model end-to-end from scratch, it is highly likely that the manager will overuse one specific expert. The typical reason is that an expert initially shows superiority because of its weight initialization or the preceding weight update, leading the manager to assign a higher weight to that expert; thus, the expert becomes more reputable due to a more frequent assignment, which eventually results in imbalanced training.

In MoQ, each expert is expected to exhibit different forecasting styles, and this is extremely challenging to accomplish if all experts are trained with the same objective function. To encourage diversity among the experts, a specific quantile loss is employed to pretrain each expert individually, and the quantile index is different for the training of each expert. As depicted in Figure 4.1, MoQ consists of four experts, and these experts are pretrained with the default quantile index 0.5, 0.7, 0.8 and 0.9 respectively; it is also feasible to customize the forecasting behaviour by either pruning specific experts or adjusting the default quantile index. Among the four experts, expert $E_{0.5}$ is the most conservative one, and the prediction becomes more aggressive as the quantile index is increased. Based on that, we can reliably obtain predictors with various forecasting styles. Once the predictions made by each expert are available, the outputs are concatenated as:

$$\hat{\mathbf{x}}^{E}_{t+1:t+h} = [\hat{\mathbf{x}}^{E_{0.5}}_{t+1:t+h}, \hat{\mathbf{x}}^{E_{0.7}}_{t+1:t+h}, \hat{\mathbf{x}}^{E_{0.8}}_{t+1:t+h}, \hat{\mathbf{x}}^{E_{0.9}}_{t+1:t+h}], \tag{4.3}$$

where $\hat{\mathbf{x}}^{E}_{t+1:t+h} \in R^{k\times N\times h}$, k is the number of experts, N is the size of mini-batch and h is the forecasting horizon. After pretraining using the quantile loss, the weights of experts are frozen and will not be updated during the training of the manager.

4.3.2 Manager

The manager is a gating function responsible for aggregating the output of individual experts. The aggregation is performed based on the recent observations of the time series to capture the local temporal behaviour, where “local" refers to the most recent p observations, with p being a hyperparameter. In this work, the manager consists of a fully connected layer and a softmax activation function. The fully connected layer extracts the local patterns of time series, and the manager calculates the softmax score among experts for each future step, which can be formulated as follows:

$$H = FC_w(\mathbf{x}_{-p:t}), \tag{4.4}$$

$$S = \text{Softmax}(\text{H}), \tag{4.5}$$

where FC_w is a fully-connected (FC) layer parameterized by the weights/biases w, $\mathbf{x}_{-p:t} \in R^{N\times p}$ is the matrix of recent observation and $S \in R^{N\times h\times k}$ is the expert score.

The dimension of the hidden vector H has to be converted from $N \times hk$ to $N \times h \times k$ before passing it to the activation function. Based on score S, the final prediction $\hat{\mathbf{x}}_{t+1:t+h}$ is made by fusing the weighted output of experts:

$$\hat{\mathbf{x}}_{t+1:t+h} = \sum_{e \in E} S^e \hat{\mathbf{x}}^e_{t+1:t+h}, \tag{4.6}$$

where E is the set of experts, $\hat{\mathbf{x}}^e_{t+1:t+h}$ and S^e are the corresponding prediction and score for expert e. To acquire the ability to use experts cooperatively, the manager is trained to minimize the Mean Absolute Error (MAE) between the fused prediction and the ground truth; the details are discussed in the next section.

4.4 Two-stage training and penalization

Training is the most important part of the MoE model; as mentioned previously, the difficulties mainly arise from two problems: (1) how to guarantee the diversity of experts; (2) how to obtain reasonable and interpretable expert assignments without overusing a specific expert. Addressing these issues necessitates more than a simple single-stage training approach; therefore, we propose a two-stage training method for MoQ to ensure the success of the training process

The first stage of training focuses on pretraining the experts, where each expert is individually trained with a specific quantile loss to promote diversity. Due to the different settings of quantile loss, the experts will not end up being too similar. Once pretraining is completed, the weights of the experts are frozen and remain unchanged during the second stage, so as to avoid undesired behavioural changes and to simplify the training of the manager. The second stage aims to train the manager to develop its ability to select and blend experts. If we want to improve the prediction of highly dynamic regions while minimizing the overall loss, selecting proper experts upon detection of different situations is very important; however, this is also tricky for the manager to learn. In the ideal case, the manager should rely more on the conservative expert for the forecasting of non-dynamic regions, and on the aggressive experts when the model needs to predict peaks. However, it is rare to obtain a trained model that truly works in this manner. In most cases, models tend to overuse the most conservative expert which is pretrained to predict the median value. Since peaks are important but relatively rare events in the dataset, the simplest

choice for minimizing the overall loss is to avoid making aggressive predictions. In this way, the model loses the ability to capture peaks, which leads to large errors in the dynamic parts while the overall loss is rather low. This process is useless in real applications because it cannot estimate the most significant part of the time series, as in the naive but apt metaphor: you will never make mistakes if you decide to do nothing. In this case, a penalization mechanism is needed against the manager's nature of being conservative. Specifically, two different penalization methods are introduced in this work.

Penalization mask

The first option is to use a mask during training. This mask acts as a penalization term in the second training stage, to penalize conservative experts and encourage the manager to assign higher weights to aggressive experts. The mask is designed as follows:

$$M = \left[\frac{1}{k}, \frac{2}{k}, \ldots, \frac{k}{k}\right], \tag{4.7}$$

where k is the number of employed experts; the mask is $M = [0.25, 0.5, 0.75, 1.0]$ when we use 4 experts. Each penalization coefficient within the mask corresponds to the expert in the corresponding position of $\hat{\mathbf{x}}^{E}_{t+1:t+h}$ (see eq. (4.3)), where the most conservative expert is penalized in the hardest way while the most aggressive expert is not penalized at all. Each expert prediction has to be multiplied by the corresponding mask coefficient before being fed into the manager; this deteriorates the conservative predictions, often making them too low to be used, and only slightly affects the aggressive predictions. In this way, predicting the median value is not the best choice anymore in order to minimize the loss, and the manager is required to learn a smarter way to fuse the results of the individual experts. This is an objective-oriented penalization method, and the mask values can be changed in order to guide the decision of the manager in the desired way, thereby controlling the behaviour of MoQ.

Penalization via addition of Gaussian noise

The second approach is to add Gaussian noise to the predictions instead of masking them. The Gaussian noise is designed to have zero mean and expert-dependent variance. For expert E_τ, the variance of its Gaussian noise is chosen as $\sigma_\tau^2 = \alpha \cdot (\frac{1}{\tau} - 1)$, where α controls the spread of the noise variance among the experts. The more conservative the expert, the higher the variance of the Gaussian noise. Compared to the penalization mask, the addition of Gaussian noise employs only one parameter and does not require the design of a penalization mask; this leads, however, to a somewhat weaker control of the model behaviour.

Use of penalization during the training process

It is important to notice that the two introduced penalization methods are only used during the training phase, whereas experts are not penalized during the validation and test phase. The reason why we penalize the predictions is to obtain a trade-off between the quality of peak prediction and overall loss, as we need to balance the behaviour of experts while avoiding having a high overall loss. With the penalization, we therefore lower the priority of using conservative predictions. Since the issue of overusing specific experts is addressed, the remaining problem is how to force the manager to extract knowledge from recent observations to guide its fusion. Indeed, we want MoQ to make aggressive predictions only when there could be an occurrence of a peak in the upcoming steps. If we apply penalization to expert predictions for all training examples in a mini-batch, this may lead to higher-than-expected predictions of non-peak parts considering that the conservative experts would be penalized all the time. To overcome this problem, we do not apply the penalization for all training samples, but only for the samples whose ground truth is much higher than the others. Specifically, for each training sample we sum the values of its ground truth, and only the samples whose sum is in the largest 10% will be penalized. The rationale is that the selected training samples have a higher possibility of having peaks in the next few steps, and the manager is forced to learn how to extract local temporal dynamics from recent observations to determine if a peak will occur or not. In this training stage, the model is trained to minimize the MAE of the final prediction. With the way of penalizing predictions for selected training samples, we introduce inductive bias in the training process, which is intended to inspire the manager to act in our

desired way. Compared to conventional models (e.g., LSTM), MoQ requires more time to train, but the general training time is still acceptable.

4.5 Mobile traffic predictor: FMLP

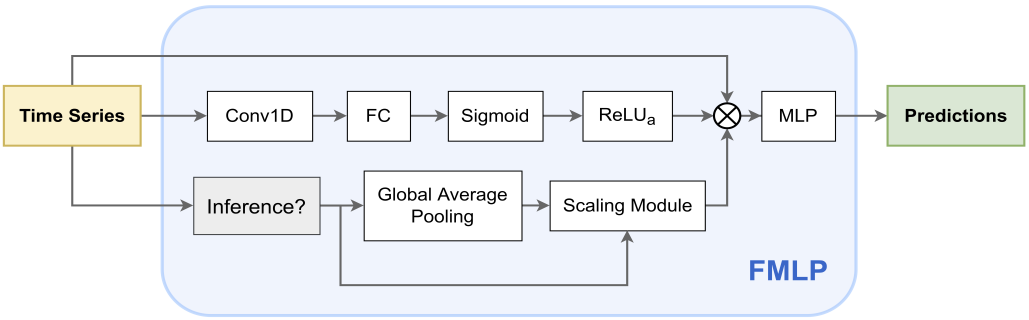


Fig. 4.2 Architecture of FMLP.

The architecture of FMLP is composed of three components, as shown in Fig. 4.2.

- an *Information Filter Module* which consists of a convolutional layer, a FC layer and two activation functions. The use of this module is to select important samples from the input and filter the potential noise, eventually creating a mask which weights each sample of the input time series.
- a *Conditional Scaling Module* which is composed of a global average pooling layer and a scaling module. The objective of this component is to perform magnitude scaling for the selected samples based on a pre-defined scaling factor; a mask is created to scale the magnitude of each sample of the input.
- a *MLP* that makes predictions by using the time series masked by the previous modules.

The information filter and conditional scaling modules can be seen together as the "data mapping" module of FMLP. The idea behind FMLP is to design a model whose peak prediction can be customized to have different aggressiveness levels. To this end, the time series is first fed into the two modules to generate two masks selecting the important samples and scaling the magnitude for part of them. By adjusting the pre-defined scaling factor, the magnitude of peak predictions can be

flexibly modified. Subsequently, the original time series is weighted by multiplying these masks and then input into the MLP for prediction generation. The details of FMLP are discussed in the following.

4.5.1 Information filter module

This module is used to assign a weight to each input sample. The convolutional layer and FC layer are employed to learn temporal patterns of time series, thereby assessing the importance of each sample. The resulting hidden vector has the same length as the input, and it is first processed by a sigmoid function to map its value to the range $(0,\ 1)$; then a parametric ReLU is applied to filter the noise, as:

$$ReLU_\gamma(x) = \begin{cases} 0, & x \leq \gamma \\ x, & x > \gamma \end{cases}, \tag{4.8}$$

where γ is a pre-defined threshold. Generally, the importance of samples is quite "sparse" as, after the sigmoid, many samples are mapped to very low values such as 0.01. These samples contribute relatively little to the prediction and may even be detrimental. Hence, the parametric $ReLU_\gamma$ is used to filter the noise and the parameter γ is set to 0.05 in the experiments.

4.5.2 Conditional scaling module

This module is used to scale the magnitude of time series samples during the training phase, but it is not employed during the inference stage. Initially, the module calculates the global average of the time series, and then this average, along with the original time series, is fed into the scaling module to generate a scale mask. Since our objective is to scale the magnitude of peak predictions without impacting the off-peak parts, not all samples require scaling. Specifically, all samples with values higher than the global average are multiplied by a pre-defined scaling factor. Thanks to the nonlinear properties of neural networks, predictions are robust to changes in input series scale; thus, even if only a few samples' magnitudes are adjusted, the model would still be optimized to make similar predictions. Therefore if we scale the magnitude during the training stage and disable scaling during the inference stage,

the magnitude of predictions will be inversely scaled without altering their shape; to increase the magnitude of peak predictions, the scaling factor should be less than 1. However, in the context of time series forecasting, one drawback of neural networks is their insensitivity to the input time series scale due to the non-linearity introduced by stacking layers [84]. If we decrease the magnitude of peak regions during the training phase, the model will still yield similar predictions during training, but the peak predictions will be increased during the inference phase if we deactivate the scaling module since the weights of layers are optimized to compensate for the magnitude reduction.

4.5.3 Basic predictor: MLP

The MLP used in FMLP is the conventional model proposed by [64], which is composed of three FC layers and ReLU functions. The MLP uses the time series weighted by the two masks to make the predictions.

4.6 Detection of traffic imbalance

4.6.1 Cluster creation and analysis

Due to the network configuration and user movement, mobile traffic often distributes among a cluster of cells in an imbalanced way, resulting in load imbalance that needs to be detected. Successfully detecting this potential load imbalance allows network operators to switch from a configuration optimised for capacity to another suited for redistributing the traffic, leading to improved performance for the congested cell. In this chapter, a cluster is defined as a set of cells which includes a reference cell (cluster center) plus a few adjacent cells (neighbours). The procedure for creating a cluster is as follows:

1. *Selection of Reference Cell*: A reference cell is more likely to be congested. We choose cells whose 0.9 quantile of its downlink usage time series is higher than 2 Erlang.

2. *Selection of Adjacent Cells*: for the selection of adjacent cells, we take into account both coverage overlap and served users overlap between the reference

and adjacent cells. We employ the handover data and consider as adjacent cells those whose served users overlap is higher than 20%, and this overlapping setting allows the redistribution of mobile users among cells; among those, the two cells having the largest coverage overlap are selected to form a cluster with the reference cell.

Once we create the clusters, for each cluster we monitor the total ratio R^t_{total}, the maximum ratio R^t_{max} and the downlink usage of the cluster. R^t_{total} is the ratio between the downlink usage of the reference cell and the total downlink usage within the cluster, and R^t_{max} is the ratio between the downlink usage of the reference cell and the maximum downlink usage among its adjacent cells. The ratios are calculated as follows:

$$R^t_{total} = \frac{\lambda_{ref,t}}{\sum_{c \in C} \lambda_{c,t}}, \tag{4.9}$$

$$R^t_{max} = \frac{\lambda_{ref,t}}{\max_{c \in A} \lambda_{c,t}}, \tag{4.10}$$

where $\lambda_{ref,t}$ is the downlink usage of the reference cell at time step t, $\lambda_{c,t}$ is the downlink usage of the cell c at time step t, C is the whole cluster and A is the set of adjacent cells. For the time step t, the reference cell is thought of as congested and requiring traffic redistribution if its downlink usage is higher than 2 Erlang and $R^t_{max} > 2$.

Algorithm 2 Naive Approach

Input: $\lambda_{ref,t}$, R^t_{max}.
Output: L.
if $R^t_{max} \geq 2$ **then**
  if $\lambda_{ref,t} \geq 2$ Erlang **then**
    $L_{t+1} = L_{t+2} = 1$
  else
    $L_{t+1} = L_{t+2} = 0$
  end if
else
  $L_{t+1} = L_{t+2} = 0$
end if

Algorithm 3 Predictive Approach

Input: $\lambda_{ref,1:t}$, $\lambda_{adj1,1:t}$ and $\lambda_{adj2,1:t}$.
Output: L
for Time step $h = t+1$ to $t+2$ **do**
  $\hat{\lambda}_{ref,h} = Predictor(\lambda_{ref,1:t})$
  $\hat{\lambda}_{adja1,h} = Predictor(\lambda_{adj1,1:t})$
  $\hat{\lambda}_{adja2,h} = Predictor(\lambda_{adj2,1:t})$
  $R^h_{max} = \frac{\hat{\lambda}_{ref,h}}{\max_{c \in A} \hat{\lambda}_{c,h}}$
  if $R^h_{max} \geq 2$ **then**
    if $\hat{\lambda}_{ref,h} \geq 2$ Erlang **then**
      $L_h = 1$
    else
      $L_h = 0$
    end if
  else
    $L_h = 0$
  end if
end for

4.6.2 Detection algorithm

To determine if there is a need to redistribute mobile traffic or not, we apply different approaches to detect the traffic imbalance within the cluster based on the downlink usage. The first approach is called the "naive" approach and it uses the latest KPI observations of the cells within the cluster to make the decision; this approach is used as the baseline by considering it is quite simple. Another option is the predictive approach which uses the predictions of downlink usage to guide the decision. Either approach provides labels of the next two steps, $L = [L_{t+1}, L_{t+2}]$, where L can be either 0 or 1, and 0 means there is no need to perform traffic redistribution.

The pseudo codes of these approaches are shown as Algorithm 2 and Algorithm 3. Furthermore, the predictive approach can be subdivided into two subcategories based on the applied forecasting models: single-model approach and multi-model approach. The single-model approach uses the same forecasting model to predict the downlink usage for both the reference cell and the adjacent cells, while the multi-model approach uses different models for handling the reference cell and the adjacent cells. For example, Model A+Model B means using Model A for the reference cell and Model B for the adjacent cells. Compared to the single-model

approach, the multi-model approach provides higher flexibility for decision-making; this is discussed in detail in Section 4.7.3.

4.7 Experiments and results

In this section, we conduct experiments on two real-world industrial datasets to evaluate the performance of the proposed approaches; both datasets are provided by Telecom Italia S.p.A. For the first dataset, the mobile data is collected from the LTE network of Torino - a major city in northern Italy, covering 100 cells where each cell monitors multiple network KPIs, such as the downlink and uplink mobile demand, percentage of call drop, the number of connected users and others; all mobile records are transmitted to mobile servers which aggregate and store them. In this work, we focus on two network KPIs, namely the downlink usage and the average number of connected users, whereas the former measures the downlink mobile demand of a cell and the latter measures the average number of users connected to a cell. Most experiments have been done using the downlink usage, and the number of connected users is only used in one experiment presented in Section 4.6. For each KPI, the dataset consists of 32300 time series (323 time series for each cell), and each time series is recorded during 14 consecutive days; the traffic profile of each cell is aggregated over 15-minute intervals. This dataset is used to train and test the forecasting performance of the proposed mobile traffic predictor. The second dataset is a much larger one; it consists of three months of observations of 2713 cells deployed in Piedmont province, Italy. In addition to KPIs, this dataset encompasses specific cell-related information including the geographical location, coverage overlapping rate with adjacent cells and user overlapping rate with adjacent cells. In experiments, it is used to perform large-scale mobile network analysis and evaluate the performance of the traffic imbalance detector.

4.7.1 Forecasting of mobile traffic peaks

For the mobile traffic predictor, our target is to predict network parameters for the next 2 steps (30 minutes); all time series of this dataset have been normalized with the standard score normalization, and the normalized time series is calculated by

first subtracting the mean value of raw time series and then dividing by the standard deviation.

Benchmarks and performance metrics

Two versions of MoQ are implemented: the MoQ penalized by mask (MoQ) and the MoQ penalized by the Gaussian noise whose variance is controlled by α ($\text{MoQ}^{\dagger}_{\alpha}$). The FMLP model is denoted as FMLP_{β} where β is the predefined scaling factor. We compare the performance of our proposed models against a set of baseline approaches covering the most popular approaches in the time series forecasting field, including MLP [64], LSTM [78], GRU [79], LSTNet [84], TCN [72], MQ-RNN [87], Transformer [88], DLinear [82], Informer [89] and Autoformer [90]. The hyperparameters of these models have been tuned through grid search based on the performance evaluated on validation set. Once the best configurations have been determined, the models have been retrained on the dataset, by merging the training and validation sets and eventually evaluated on the test set. All experiments are conducted on a computer with Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz, 64 GB memory and a single Nvidia TITAN X Pascal 12GB GPU. The operating system is Linux 4.15.0-143-generic, and the model is implemented in Python 3.8.8 with Pytorch 1.7.0.

To evaluate the performance, five metrics are used. Initially, we quantify the overall performance of these models in terms of MAE and MSE (Mean Squared Error). Since MAE and MSE do not properly capture the ability of a model to predict peaks in the time series, which is instead very important in our target application, we also employ classification metrics to evaluate the ability of models to predict peaks. Specifically, for the downlink usage the peak part is defined as the elements of a tensor whose value is higher than a threshold Q defined as a given quantile of this feature. In this setting, a mobile traffic predictor that forecasts downlink usage two time steps ahead is employed for binary peak classification, in that if the predicted downlink usage has a value greater than or equal to the predefined threshold, the prediction is seen as positive (peak) and a peak is considered as being predicted correctly if the ground truth has the same label (positive). In our experiments, the quantile index is defined as 0.95. Based on this, each ground truth element is either positive (peak) or negative (non-peak), and a model is assessed on the basis of its ability to correctly classify peaks in terms of the sensitivity metric, also

Models	MAE	MSE	Accuracy	Sensitivity	#Parameters
MLP	0.301	0.297	67.5%	36.2%	188k
LSTM	0.295	0.303	67.7%	36.5%	265k
GRU	0.286	0.298	66.1%	33.1%	50k
LSTNet	**0.282**	**0.287**	67.5%	36.2%	102k
TCN	0.298	0.326	57.0%	14.2%	61k
MQ-RNN	0.290	0.299	66.2%	33.4%	141k
Transformer	0.332	0.342	69.1%	39.3%	1.12M
DLinear	0.295	0.293	66.3%	33.7%	**5k**
Informer	0.343	0.427	58.7%	18.3%	660k
Autoformer	0.411	0.481	56.3%	13.6%	1.14M
MoQ	0.310	0.329	**77.7%**	**58.5%**	409k
$\text{MoQ}_2^\dagger$	0.297	0.297	75.2%	52.6%	409k
$\text{MoQ}_4^\dagger$	0.306	0.310	**77.7%**	58.2%	409k
$\text{FMLP}_{0.7}$	0.301	0.312	75.5%	53.5%	189k

Table 4.1 Performance comparison of different models for mobile traffic prediction and peak classification; k and M are the shorthand notation for Thousand and Million.

known as recall, i.e. the number of correctly identified peaks among the retrieved peaks. Although sensitivity is the most important metric for the target application, we also need to verify that a model does not overestimate a target frequently by generating a lot of false alarms, so the average classification accuracy among classes is also used to quantify the general performance of predictive classification. Besides these metrics, the number of parameters (#Parameters) is also used to evaluate how many parameters are included in the models, which is a good proxy of the model complexity.

Results

We perform forecasting on the test set of the mobile traffic dataset; the performance of the models is reported in Table 4.1. For this dataset, we observe that RNN-based models obtain better performance compared to the others. LSTNet has the lowest MAE and MSE and it is the best model if we only consider the overall loss. The performance of Autoformer is the worst among the baselines, its peak prediction performance is much worse than the others and the forecasting performance is quite poor. Compared with the selected baselines, the proposed MoQ models obtain much

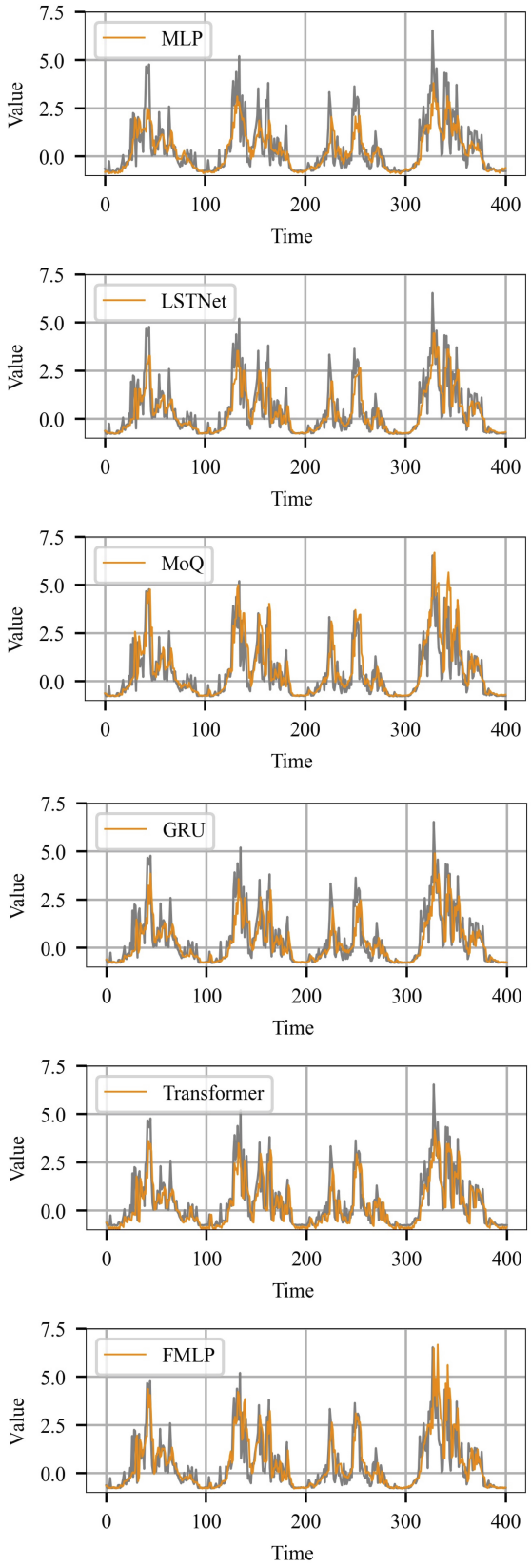


Fig. 4.3 Predictions of downlink usage. The penalization mask of MoQ is employed in the plot and the scaling factor of FMLP is set to 0.7.

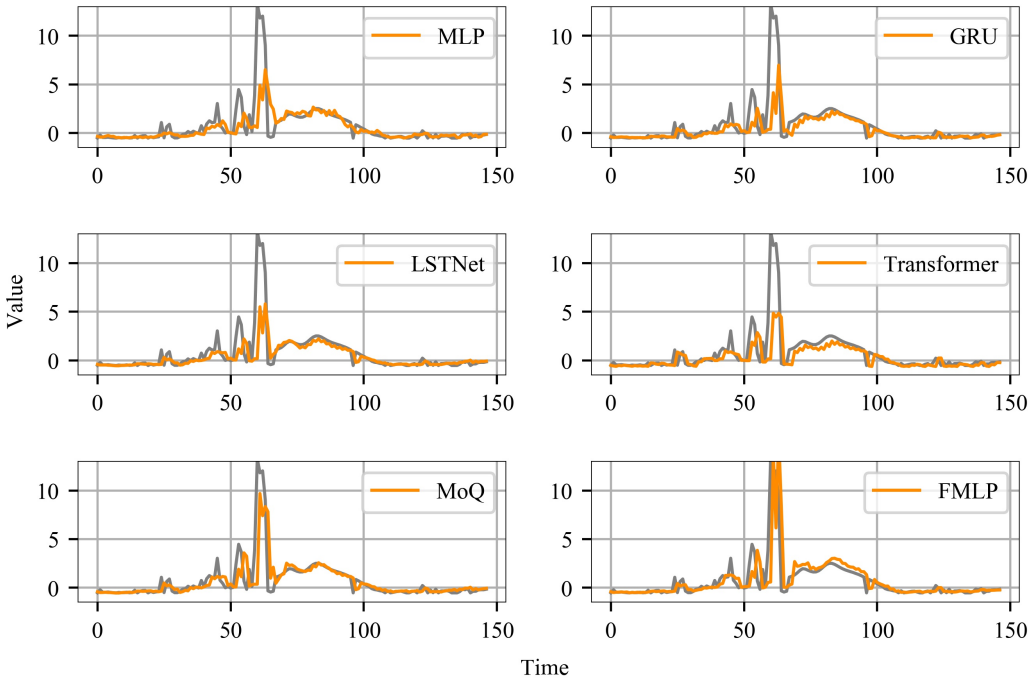


Fig. 4.4 Peak predictions of downlink usage. The penalization mask of MoQ is employed in the plot and the scaling factor of FMLP is set to 0.7.

higher sensitivity. If we compare the performance between the mask version MoQ (MoQ) and the Gaussian noise versions (MoQ$^{\dagger}_{\alpha}$), the difference between them is quite small, and both of them provide a significant improvement of peak predictions. Among these MoQ models, MoQ has the highest sensitivity whose value is 19.2% higher than the highest sensitivity of baselines (39.3%), which means this model is much more capable of predicting potential upcoming peaks; MoQ also has the highest classification accuracy, showing that the proposed model is able to perform forecasting based on the recent trend which better reflects the traffic pattern. The price to be paid for very good sensitivity is that its MAE and MSE are slightly higher than those of LSTNet, though still close to those achieved by the best methods. Comparing MoQ$^{\dagger}_{4}$ and MoQ$^{\dagger}_{2}$, we observe that the MAE and MSE are reduced by decreasing the value of α while obtaining slightly lower sensitivity and classification accuracy; by modifying the value of α, it is possible to make a trade-off between peak prediction and overall performance. Compared to MoQ, FMLP obtains slightly worse forecasting performance, but its performance is still much better than the baselines for the task of mobile traffic peak forecasting. If we compare the complexity of the models, we can find that MoQ is not a very heavy model even though it is built based

on the expert system, and it is still lighter than some baseline approaches such as the transformer-based models; to have a faster and lighter version of MoQ, FMLP would be a good candidate as its model size is smaller.

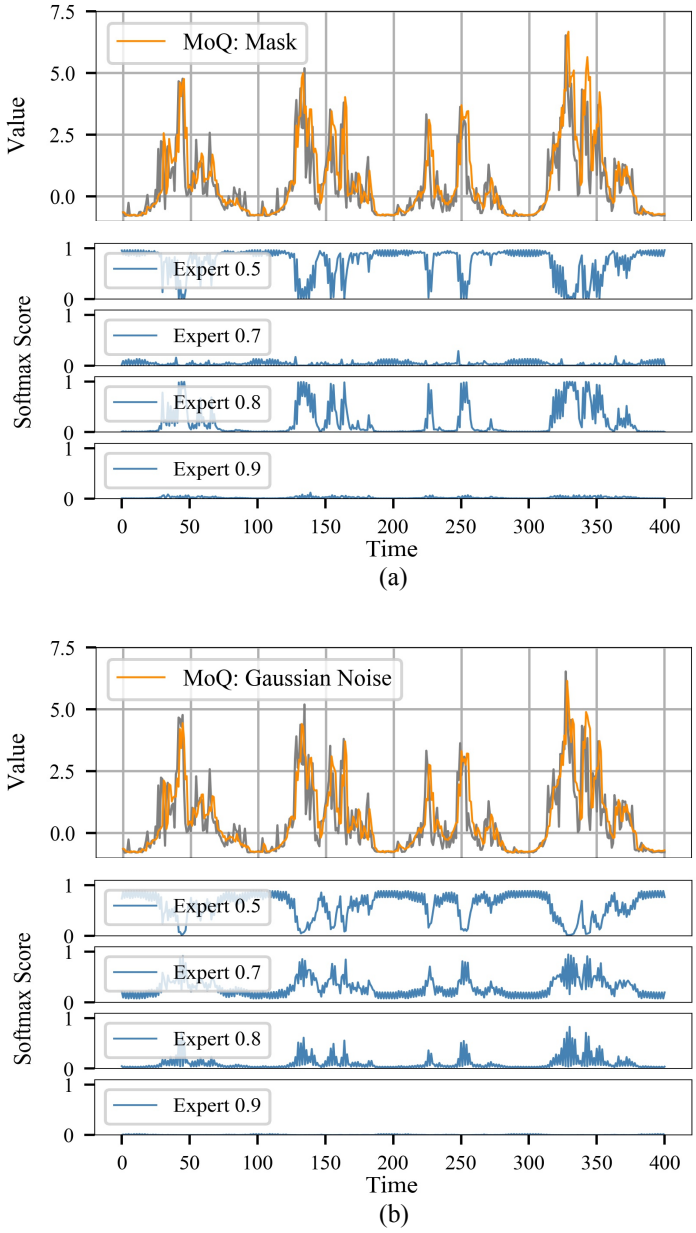


Fig. 4.5 Visualization of the cooperation between experts, the expert scores used to fuse the predictions are visualized. (a) MoQ (mask version); (b) MoQ (Gaussian noise version).

To better understand the behaviour of different architectures, we select several models and visualize their forecasting. Figure 4.3 and Figure 4.4 illustrate the general traffic predictions and the peak predictions respectively, where Figure 4.3 refers to the general pattern of downlink usage and Figure 4.4 refers to a bursty increase of downlink usage in mobile networks. In Figure 4.3, we can observe that the predictions made by MoQ and FMLP can better follow the trend and peaks of mobile traffic; this behaviour is more obvious in Figure 4.4, where MoQ and FMLP show their superiority in peak prediction of mobile traffic; in this case, the predicted peak is very close to the ground truth and none of benchmarks is capable of capturing the usage pattern related to the peak in an effective way.

4.7.2 Analysis of the characteristics of mobile traffic predictors

MoQ

Besides the advantage of achieving better peak prediction, MoQ also has good interpretability. As mentioned in previous sections, MoQ relies on the cooperation mechanism of experts, where the prediction benefits from switching among predictors with various forecasting styles. To study the contribution of each expert, in Figure 4.5 we visualize the predictions and the corresponding softmax scores of the four different experts: expert $E_{0.5}$, expert $E_{0.7}$, expert $E_{0.8}$ and expert $E_{0.9}$. Among these experts, expert $E_{0.5}$ and expert $E_{0.7}$ are the experts making relatively conservative predictions. In Figure 4.5(a) we observe that the conservative expert dominates the final prediction for the non-peak part of time series, whereas the prediction style becomes aggressive once our model detects strong temporal dynamics from recent observations. If we focus on the strong oscillating area of this sequence, frequent switching between expert $E_{0.5}$ and expert $E_{0.8}$ can be seen. If we compare Figure 4.5(a) and Figure 4.5(b), the learned cooperation pattern is not the same by applying different penalization methods during training, but both of the methods result in similar results; this provides additional flexibility for adjusting the forecasting strategy. From this point of view, MoQ shows great interpretability, and the analysis of the behaviour of experts can be used to fine-tune the architecture and adapt it to different telecommunication applications.

FMLP

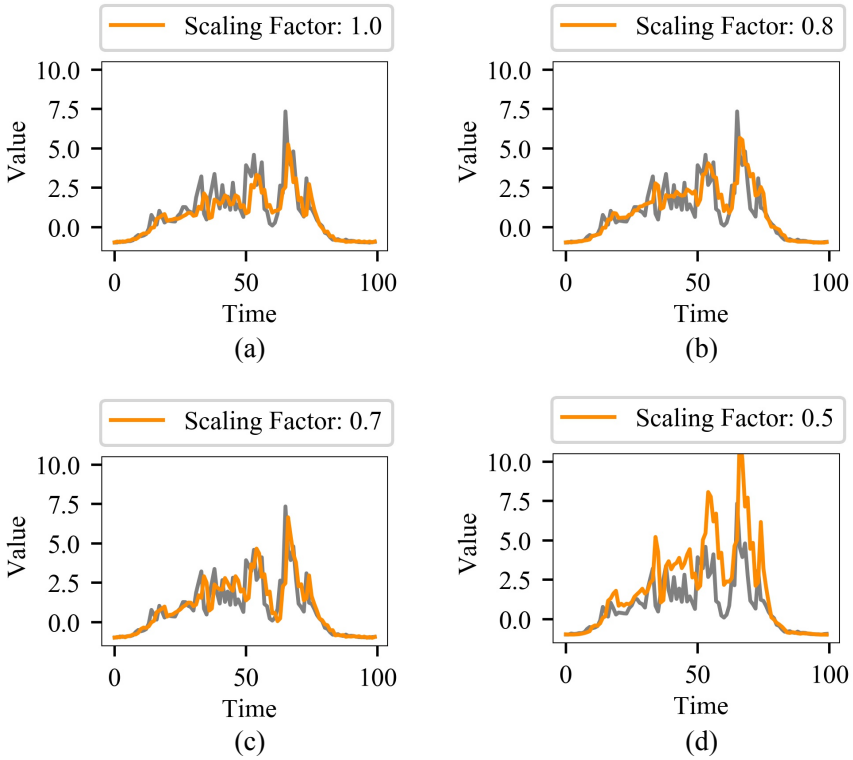


Fig. 4.6 Peak predictions of cell B made by FMLP with respect to different settings of scaling factor, the gray line represents the ground truth. (a) scaling factor = 1.0; (b) scaling factor = 0.8; (c) scaling factor = 0.7; (d) scaling factor = 0.5.

As we discussed in the previous section, the behaviour of FMLP can be adjusted by changing the value of the scaling factor, which provides us with the flexibility of customizing the aggressiveness of forecasting based on the needs of applications; here we discuss the impact of changing scaling factor to the peak predictions. In Figure 4.6, we visualize the predictions of another cell made by FMLP; four scaling factors are used to train FMLP: 1.0, 0.8, 0.7 and 0.5. When the scaling factor is set to 1.0, the difference between FMLP and the conventional MLP is minor; hence the predictions in Fig. 4.6(a) are equivalent to the predictions made by MLP. If we decrease the scaling factor from 1.0 to lower values, we can observe that the predictions of the peak parts have a higher magnitude, and that makes the predictions more similar to the real cases. Every time we decrease the scaling factor, what we do is create a smooth transition from conservative peak predictions to aggressive

Models	MAE	MSE	Accuracy	Sensitivity
Scaling Factor=1.0	**0.289**	0.309	63.2%	27.2%
Scaling Factor=0.8	0.296	**0.294**	73.3%	48.6%
Scaling Factor=0.7	0.301	0.312	75.5%	53.5%
Scaling Factor=0.6	0.381	0.528	**84.4%**	**74.6%**
Scaling Factor=0.5	0.403	0.608	83.9%	74.0%

Table 4.2 Performance comparison of FMLPs trained with different scaling factors.

peak predictions, and this process is very flexible and can be controlled based on the operational strategy of network operators. Table 4.2 shows the performance of different FMLPs; generally, a lower scaling factor leads to better peak predictions and a slightly higher average loss.

4.7.3 Predictive detection of potential traffic imbalance

Mobile traffic analysis

In our approach, the predictive detection algorithm is applied to identify the potential traffic imbalance in a reference cell once we have the predictions. To investigate if there is a benefit of using the predictive approach, we create clusters and evaluate the performance of different approaches on them. We consider two types of cells, i.e. 800MHz and 1800MHz cells; the cluster is composed of cells in the same band, following the procedure discussed in Section 4.6. In the analysis, we use three-month observations of 2713 cells (1059 800MHz cells and 1654 1800MHz cells) and identify 417 reference cells which have high downlink usage.

	Congested	Non-Congested	Total
800MHz Cells	120 (11.3%)	939 (88.7%)	1059
1800MHz Cells	231 (14.0%)	1423 (86.0%)	1654

Table 4.3 Overview of clusters: congested and non-congested.

According to Table 4.3, at least 11.3% of 800MHz cells and 14.0% of 1800MHz cells are affected by traffic congestion events after evaluating the downlink usage and the maximum ratio, indicating that traffic redistribution would be very useful in a real-world scenario. To better understand the traffic imbalance issue, we study the

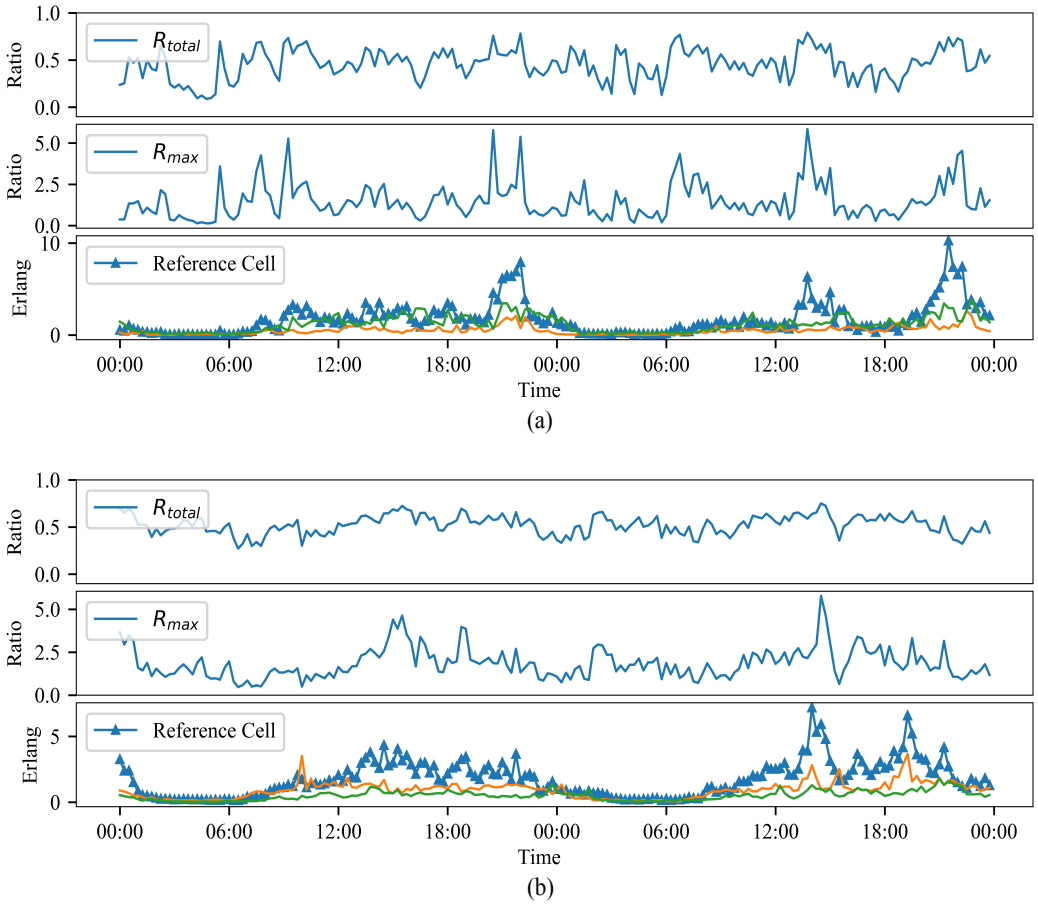


Fig. 4.7 Traffic distribution within the cluster in two consecutive days. The marked line is the downlink usage of the reference cell and the unmarked lines are the downlink usage of its adjacent cells within the cluster. (a) cluster A (800MHz); (b) cluster B (1800MHz).

behaviour of the clusters in detail. Figure 4.7 illustrates the metrics of two clusters in two consecutive days; as we can observe, the downlink usage of the reference cell is much higher than the downlink usage of its adjacent cells in some time slots. In Figure 4.7(a), congestion can be observed around 9 PM; the congestion pattern is not always periodic as the congestion event is observed in the afternoon without having been observed in the previous day. According to the figure, it is obvious that mobile traffic is distributed in a relatively balanced way most of the time. However, for certain time slots, the reference cell carries much more traffic than the others which would potentially affect the QoS of the mobile users assigned to it. In this case, the prediction of the traffic imbalance would help mobile traffic management.

Detection of traffic imbalance

To compare the performance of different approaches, we set the various prediction models to perform single-model and multi-model approaches. For the single-model predictive approach, we use five different DL models to predict the downlink usage of the future steps, including MLP, GRU, LSTNet, Transformer, MoQ and FMLP (with two different scaling factors); for the multi-model approach, we use different combinations of models including LSTNet+GRU, MoQ+GRU, MoQ+LSTNet, GRU+MoQ and FMLP+GRU. The nomenclature for the multi-model approach is of the kind "$\text{Model}_{\text{ref}}$ + $\text{Model}_{\text{adj}}$", for example MoQ+GRU means using MoQ for the reference cell and GRU for the adjacent cells. By using different predictive approaches, it is feasible to evaluate how peak forecasting performance affects the prediction performance of mobile traffic imbalance. To evaluate the classification performance, four metrics are calculated:

- *Balanced Accuracy*: the classification accuracy whose value equals to the mean value of the classification accuracy of each class.
- *Accuracy (Non-Congested)*: the classification accuracy of the non-congested class.
- *Accuracy (Congested)*: the classification accuracy of the congested class.
- *F-score*: this is the harmonic mean of precision and recall which evaluates the overall performance of a classifier. Its value ranges between 0 and 1, where 1 indicates a perfect classifier.

In the experiments, 20 clusters (10 at 800MHz and 10 at 1800MHz) are employed to compare the performance of different approaches. For each cell within a cluster, we first normalize its observations of three months and split them into many sub-series; each sub-series contains 1344 records corresponding to the observations of two weeks. For each sequence, the forecasting model first predicts the normalized value of the downlink usage for the next two steps, then the real value (in Erlang) of the predicted downlink usage is obtained by denormalizing the predictions using the corresponding mean and standard deviation, and it is used by the predictive approach for decision making; the ground truth of the predictive classification is generated using the future downlink usage (in Erlang) of the reference cell and the adjacent

cells, and the details can be found in Section 4.6.1. The results are shown in Table 4.4.

		Balanced Accuracy	Accuracy (Non-Congested)	Accuracy (Congested)	F-score
	Naive Approach	85.1%	91.9%	78.3%	0.787
	Predictive Approach				
Single-M	MLP-based	86.9%	92.4%	81.5%	0.810
	GRU-based	86.7%	92.4%	81.1%	0.808
	Transformer-based	85.1%	92.2%	78.2%	0.788
	LSTNet-based	87.3%	92.1%	82.4%	0.813
	MoQ-based	87.4%	91.4%	83.3%	0.811
	FMLP-based (scaling factor=0.6)	86.5%	90.5%	82.5%	0.797
	FMLP-based (scaling factor=0.7)	87.1%	91.9%	82.2%	0.809
Multi-M	LSTNet + GRU	87.5%	91.5%	83.5%	**0.814**
	MoQ + LSTNet	88.1%	90.1%	86.0%	0.813
	MoQ + GRU	**88.2%**	89.4%	87.0%	0.811
	GRU + MoQ	84.9%	**93.8%**	76.0%	0.791
	FMLP (scaling factor=0.6) + GRU	88.0%	87.4%	**88.6%**	0.801
	FMLP (scaling factor=0.7) + GRU	87.7%	90.8%	84.7%	0.812

Table 4.4 Comparison of different approaches; Single-M and Multi-M are the abbreviations for single-model approach and multi-model approach.

In Table 4.4, the best two models are the multi-model approach MoQ+GRU and FMLP (scaling factor=0.6)+GRU as they have much higher accuracy of detecting the congestion events while obtaining good overall performance. For instance, compared to the naive approach, MoQ+GRU achieves 3.1% higher average classification accuracy and 8.7% higher accuracy in classifying the potential congestion; even though the use of prediction models introduces additional complexity (Table 4.1), the cost is still acceptable for mobile operators considering the improvement of congestion detection is significant.

From a global point of view, predictive approaches are better than the naive approach, especially for detecting future mobile traffic imbalances. Among the single-model approaches, the MoQ-based approach is the best one as we care more about the sensitivity to the potential congestion, and its performance is slightly better than the LSTNet-based approach. The performance of the single model FMLP-based approach is similar to LSTNet-based and MoQ-based approaches when the scaling factor is set to 0.7; when we apply the multi-model approach with FMLP, the performance of FMLP+GRU is better if the scaling factor equals to 0.6. As we discussed, the ability of MoQ and FMLP to predict peaks makes them very good candidates for predicting the rapid increment of mobile demand. However, using only the single model could be sub-optimal in the scenario of traffic redistribution. Due

Feature	Balanced Accuracy	Accuracy (Non-Congested)	Accuracy (Congested)	F-score
Native Approach				
Downlink usage	**85.1%**	**91.9%**	**78.3%**	**0.787**
#Connected users	67.9%	69.1%	66.6%	0.540
MLP-based Approach				
Downlink usage	**86.9%**	**92.4%**	**81.5%**	**0.810**
#Connected users	67.6%	68.5%	67.4%	0.541

Table 4.5 Comparison of different features.

to the fact that MoQ and FMLP are more capable of making aggressive predictions, they tend to make higher predictions for both the reference cell and the adjacent cells; because the maximum ratio measures the relative difference between the downlink usage of the reference cell and its adjacent cells, in this case we can purposely obtain a higher ratio by using a more conservative predictor to predict the adjacent cells, which makes the classifier more sensitive to the future load imbalance. As a result, the approaches MoQ+GRU and FMLP+GRU show great performance in detecting the traffic imbalance. Besides improving the ability to detect traffic imbalance, we can also make the classifier focus more on the balanced case. The approach GRU+MoQ uses a conservative predictor for the reference cell and a more aggressive predictor for the adjacent cells, which leads to the highest classification accuracy on the non-congested class (93.8%), but with much worse performance on other metrics. Compared to the naive approach and the single-model approach, the multi-model approach shows better performance and provides higher flexibility, as it can be adjusted based on the needs of different operating strategies of network operators. Considering the results of both single-model solutions and multi-model solutions, MoQ is seen as a better choice compared to FMLP; even though FMLP can achieve slightly better results in multi-model cases (FMLP 0.6+GRU), the low scaling factor would lead to much worse general forecasting performance which is not desired (Table 4.2). In this case, FMLP is a good candidate designed for applications which have higher requirements of computation speed and model complexity.

Besides the comparison of algorithms, our attention should also be focused on the choice of the network KPIs used to perform the decision-making. Among the network KPIs, the average number of connected users is another important variable which reflects the mobile demand indirectly. Compared to the downlink usage, this variable exhibits smaller variation over time which makes it much easier to predict,

and it is interesting to study if the predictive approach performs better using as input the number of connected users with respect to the downlink usage. Table 4.5 compares the performance of the approaches using different variables. In the table, it is obvious that the average number of connected users is not a good choice for guiding the decision of traffic redistribution, as it cannot measure the mobile traffic correctly. The reason behind the result is simple: the mobile traffic is not always proportional to the number of connected users considering some users would not generate too much traffic; in this case, even though a cell has a high number of connected users, there is no guarantee that the cell also has a high downlink demand. From this point of view, using the downlink usage would be the better choice.

4.8 Conclusions

In this chapter, we have explored the topics of how to predict mobile traffic peaks and mobile traffic imbalances accurately, which have been rarely studied in the literature. To address these issues, we have proposed a prediction-based approach to detect potential mobile traffic imbalances among cells. The proposed approach is composed of a mobile traffic predictor (selected from MoQ and FMLP) and a traffic imbalance detection algorithm.

Compared to other forecasting models, MoQ and FMLP can make much more accurate peak predictions, where MoQ relies on a cooperation mechanism between experts and FMLP uses a conditional scaling module. MoQ stands out for its exceptional forecasting performance and interpretability, despite being a somewhat "heavier" model from the computational standpoint. With the aim to reduce the computational load, we also present a lightweight alternative, i.e. the FMLP model. Extensive experiments are carried out on a real-world dataset, and the results have proved the effectiveness of our approaches; regarding the prediction of mobile traffic peaks, MoQ and FMLP have improved the sensitivity (the peak prediction accuracy) by approximately 19% compared to the highest sensitivity of baseline approaches (39.3%), where the improvement is roughly as large as 50%.

For traffic imbalance detection, we propose two methods including the single-model predictive approach and a multi-model predictive approach. We first perform the large-scale mobile traffic analysis by using more than 2700 cells deployed in northern Italy, then the experiments are conducted on real-world datasets to evaluate

the performance of the proposed detection approach. Compared to the baseline approach, our proposed predictive approaches significantly improve the accuracy of congestion prediction, where the best multi-model approach improves the predictive classification accuracy of congestion (the imbalance detection accuracy) by 10.3% with respect to the naive approach which uses recent observations. We also explore whether the selection of predicted network KPIs matters or not, and the results show that the downlink usage can better reflect the real usage case of mobile networks.

For future work, there are several potential research directions. First, it would be interesting to integrate the proposed approach with mobile network optimization approaches such as MLB and CCO, which would lead to more efficient network optimization and traffic handover for mobile traffic congestion. The proposed mobile traffic peak forecasting models can also be used to support the predictive maintenance of mobile networks, where the mobile operators can schedule maintenance activities during periods of low expected mobile demand to minimize service disruptions. For the predictors themselves, one potential study is to explore how to obtain good peak forecasting performance along with very good general forecasting performance; we believe that there is still room to improve the forecasting performance of the proposed solutions.

In this chapter, we focus on predicting mobile traffic peaks where how to improve the general forecasting performance has not been discussed. In Chapter 5, we will present the solutions which can be used to obtain excellent general forecasting performance under different constraints.

Chapter 5

Cost-effective mobile traffic forecasting

In the previous chapter, we discussed the importance of making accurate predictions of mobile traffic peaks; two DL models have been proposed to improve peak forecasting performance while maintaining a relatively low MAE and MSE. In this chapter, we keep presenting the works related to mobile traffic forecasting - this time we focus on improving the general forecasting performance considering deployment constraints instead of the peak case. As presented in Chapter 2, many works have studied how to improve mobile traffic forecasting; meanwhile, the difficulties which may be encountered during deployment are usually ignored; for mobile operators, how to reduce the cost of training and deploying forecasting models is also very crucial because of the revenue-driven nature. In this chapter, the material appeared in my publications [189, 190].

5.1 Forecasting with different constraints

There are many applications which require good accuracy of mobile traffic forecasting, such as the base station sleeping [191], the admission control [192] and the resource allocation and scheduling [193, 194]. To obtain better mobile traffic forecasting, many popular models in the time series forecasting field are used to perform mobile traffic forecasting [124, 195]. Even though these models perform generally well in time series forecasting, their forecasting performance on mobile

traffic is quite limited because of the nature of mobile traffic time series. Predicting mobile demand accurately is never an easy task due to the characteristics of mobile traffic over time: one can observe strong periodicity as the wireless demand closely follows human daily activities, whereas traffic is also affected by unexpected events (social events, etc.) adding variability, resulting in complex temporal dynamics. The temporal pattern of mobile demand is a mixture of the short-term periodic pattern, long-term periodic pattern and the recent trend affected by unknown factors, which results in a very complicated mobile traffic pattern which is difficult to analyze and predict. Although the current DL models are powerful in modelling the temporal correlation within the sequence, it is still hard to capture this complex pattern.

In practice, improving mobile traffic forecasting is an even more tricky problem as there is concern about the constraints of the model deployment. On the side of mobile operators, the main constraints that have to be taken into account are the model size and the training time, which have not been explored thoroughly in literature; these constraints may not exist at the same time, and this depends on the specific target applications the models are applying to. The first constraint is about the model size of DL models, in many cases the DL model requires a large number of parameters to improve the model performance, which is a critical issue for devices with limited memory. Under the mobile edge computing scenario, applications and services would be deployed on mobile or embedded devices close to the mobile edge to reduce the content and service delivery delay, thereby enhancing the overall use experience for end users. [116]; large model cannot be deployed to these devices by considering the model consumes a significant amount of memory. In this case, it is desired to have a lightweight mobile traffic forecasting model which can obtain good predictions with fewer model parameters. Another issue is related to the training time of DL models; among DL architectures, RNN-based models are very well suited to time series forecasting, and this is also true for mobile traffic forecasting. However, their training process is rather time-consuming. In RNNs, the calculation of the current hidden state is dependent on the preceding states, which means the process cannot be parallelized. Considering to apply RNN-based models to predict mobile traffic time series, this problem is not negligible since network operators typically retrain the predictor every week in order to capture the latest trend of traffic demand. Generally, a city-wide LTE network can easily cover thousands of cells, and the total amount of cells would go for hundreds of thousands considering all the areas managed by the mobile operators; when there are so many cells in the system, the

training of forecasting model would be very expensive and consume a considerable amount of energy, and the situation is more critical when the forecasting model itself has a high training cost. As mobile operators are seeking higher profit, it is always desired to reduce the training cost without losing the accuracy of forecasting.

To address these challenges, we propose a novel DL model individually for each constraint, where Temporal Dynamics Aware Network (TDANet) and Variability-Enhanced Network (VEN) are used to overcome the model size and training time constraints respectively. The main contributions of this chapter are summarized as follows:

- We propose a novel model called TDANet, which improves the mobile traffic forecasting performance by leveraging both the periodic dependencies and the recent changing trend.
- We propose a novel model called VEN, which adds different degrees of variability to a time series, allowing the network to make improved mobile traffic predictions along with a very low training time. In order to reduce the training time as much as possible, the design of VEN uses only FC layers. This design makes the network able to obtain both excellent forecasting performance and very fast training speed. According to the results, its predictions have state-of-the-art accuracy while reducing training time by approximately 99%.

5.2 Problem formulation

In this chapter, our goal is to make short-term time series predictions of the downlink usage traffic using a DL model trained based on past observations. Assuming we want to predict the future values of downlink mobile traffic with w forecast horizons, a univariate time series is represented as a vector $\mathbf{x}_{1:t} = [x_1, x_2, ..., x_t] \in R^t$ which consists of past observations of the target mobile network KPI, where x_i is the record collected at time step i and t is the length of sequence; the problem can be formulated as:

$$\hat{\mathbf{x}}_{t+1:t+w} = f(\mathbf{x}_{1:t}), \tag{5.1}$$

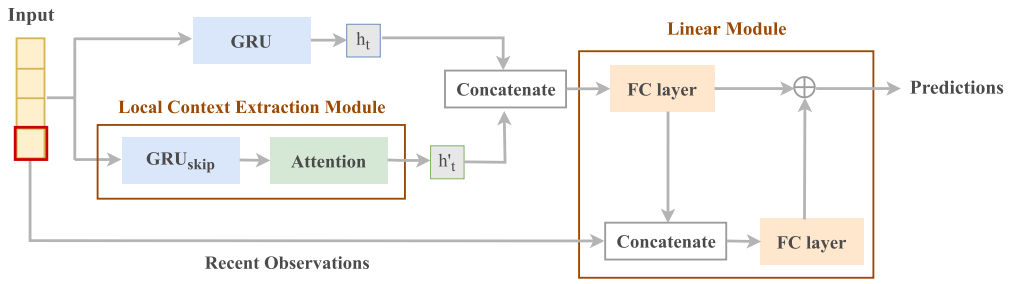


Fig. 5.1 Architecture of TDANet, where h_t and h'_t are the hidden vectors extracted by GRU and the local context extraction module respectively; the details of the local context extraction module are illustrated in Figure 5.2.

where $\hat{\mathbf{x}}_{t+1:t+w} \in R^w$ is the vector of predictions from time $t+1$ up to time $t+w$, and $f(\cdot)$ is the employed DL predictor.

5.3 TDANet

TDANet is a superlight model proposed to make accurate mobile traffic predictions, allowing mobile operators to deploy the model with the minimum requirement of computational memory. Figure 5.1 presents the architecture of TDANet. For short-term mobile traffic forecasting, the model is designed for learning the complex temporal dynamics of mobile traffic time series employing fewer parameters, which allows us to obtain accurate mobile traffic forecasting with low computational cost. To perform time series forecasting, TDANet first extracts the global and the local temporal patterns using Gated Recurrent Unit (GRU) [79] and the local context extraction module separately, then the extracted hidden vectors are concatenated and fed into the linear module to make predictions; within the linear module, the first FC layer is responsible for making initial predictions, and these predictions are concatenated with the most recent observations; the concatenated vector is used by the second FC layer to generate linear offsets, finally the final predictions are created by adding initial predictions and linear offsets element-wise. In this section, we present the layers of TDANet in detail.

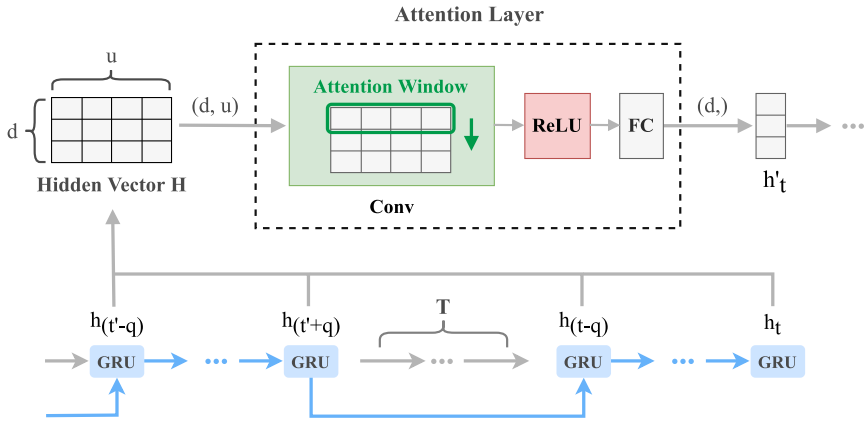


Fig. 5.2 The local context extraction module. t is the most recent time step, t' is the time step one or more skip periods away from t, T is the skip period and q defines the size of the local context. GRU_{Skip} first calculates the hidden state of the non-skipped time steps, then all the hidden states are fed into the attention layer to obtain the weighted average hidden state h'_t.

5.3.1 Recurrent component and local context extraction module

Once TDANet receives the input, the time series is first fed into the recurrent component GRU and the local context extraction module which is composed of the skip-gated Recurrent Unit (GRU_{Skip}) and an attention layer. The objective of the components is to extract the temporal pattern of mobile traffic, where GRU focuses on learning the global temporal pattern and the local context extraction module focuses on learning the local temporal pattern. Based on the use of GRU, the model obtains a hidden state h_t extracted by GRU at time step t, and this vector summarizes the coarse global information of the sequence.

Compared to the conventional Recurrent Neural Network (RNN), GRU and LSTM [78] are designed to model the long-term dependencies of time series and address the vanishing gradient problem; by applying the gate mechanism, they can better capture the correlation among samples. Even though the gate mechanism counters vanishing gradients, it is still difficult to learn the temporal dependencies between far-away samples when the time series is rather long; if the model cannot capture such long-term dependencies, there is a potential risk of losing the ability to capture periodic patterns which limits the performance in time series forecasting. To alleviate this issue, we propose to use GRU_{Skip} to learn the local patterns in preceding time steps. In practice, in several domains it is often the case that a time series exhibits strong periodicity, e.g. mobile traffic time series or outdoor

temperature time series; in this case, a common approach to predict future values is to refer to not only the recent samples but also the samples at past instants $t - nT$ shifted by an integer number of periods T, as the periodic behaviour would help the model estimate the future values.

Following this idea, GRU_{Skip} is designed to enhance the model's use of long-term dependencies; the architecture of GRU_{Skip} is presented in Figure 5.2. Instead of feeding all input samples to the recurrent component, we only use the samples which can represent a so-called *local context*, i.e., the samples just before and after the target instants $t' = t - nT$ with $n = 0, 1, \ldots,$ and T is referred to as the "skip period". By considering that the mobile traffic demand is strongly related to human activities, the periodic pattern could appear multiple time steps earlier or later than the last observed time step t which leads to a shift on the time axis; under this scenario, we prefer to focus on the samples within a predefined window centered around the time steps t', instead of focusing on just the samples at the time steps t'. For example, we would define a window which only includes the past samples from 9 AM to 11 AM if we want to predict the mobile traffic at 10 AM, and this predefined window is called the local context of this specific time step. In the GRU_{Skip} module, we have to define two hyper-parameters: window size q and skip period T; the former defines the size of the local context and the latter defines how many time steps the recurrent component needs to skip. In experiments, the window size is set to 4 which includes the samples one hour before and after the given time step, and the skip period is selected as 96, i.e. the number of samples corresponding to one day which is an obviously typical periodicity for mobile phone usage. For a given time series, the output of GRU_{Skip} is a collection of hidden states $H \in R^{d \times u}$ where d is the hidden dimension of GRU_{Skip} and u is the number of extracted hidden states. A similar design has also been used in Section 5.4 where we focus on the samples of some specific time slots.

Once the model computes the hidden vector H, the vector is passed to the attention layer to calculate the weighted average h'_t. The attention layer is composed of a convolutional layer (Conv), a rectified linear unit (ReLU) activation function and a FC layer. The reason why we calculate the weighted average h'_t instead of using the hidden state h_t is the following: even though GRU_{Skip} focuses on a specific time slot of each day, there is no guarantee that all past days are equally important, and multiple types of periodic patterns may be found in the time series, such as the daily pattern and the weekly pattern, whose combined effect is not easy to model.

Besides the periodic patterns, the most recent values of the time series are also very representative as they reflect the most recent dynamics of the mobile users' behaviour. Therefore, the future values may follow a complex periodic pattern while also depending on the most recent activities of the users. The hidden state h_t focuses more on the recent trend and downplays the periodic component. To optimally weight all factors in the forecast, the attention layer assigns different weights to the extracted hidden states and the weighted average h'_t provides an aggregated hidden state that can better summarize the behaviour of mobile traffic at the target time slot. To achieve this, the attention layer uses a Conv layer as the sliding window to extract the pattern of hidden states; note that the length of the attention window is u and the height is 1, and we slide the window along the hidden dimension d; the vector extracted by the window is processed by ReLU and passed to the FC layer to calculate the weighted hidden state. This process is formulated as:

$$h'_{s,t} = \mathrm{FC}(\mathrm{ReLU}(\mathrm{Conv}(\mathrm{H_{s,1:u}}))) \tag{5.2}$$

where s is the current sliding position on the hidden dimension, $h'_{s,t}$ is the weighted average of $H_{s,1:u}$ at the sth position of hidden dimension. Eventually, we create a new vector by concatenating the weighted hidden state h'_t extracted by $\mathrm{GRU_{Skip}}$ and the hidden state h_t extracted by GRU, and this new vector is fed into the linear module to make the predictions.

5.3.2 Linear module

The last two layers of TDANet are two FC layers which are used to make the final predictions. Compared to the models built based on feedforward neural networks, the RNN-based models show their practical superiority at learning temporal patterns of time series because the strong non-linearity allows them to have a larger search space and better fit to the data distribution. However, their non-linear nature raises an issue that the RNN-based models are not sensitive to the magnitude change of the input; when an increasing trend or a decreasing trend is observed recently, these models would slowly adapt to the change resulting in a delay in making predictions following the recent trend, which limits the time series forecasting performance. To improve the model's ability to capture recent trends, we add linear offsets to the predictions to make them more sensitive to the input scale. To do this, first the

concatenated vector learned by the recurrent components is fed into the top FC layer to make the initial predictions $o \in R^w$, then the initial predictions o and the most recent p observations are fed into the bottom FC layer to calculate linear offsets, and the final predictions are obtained by computing the element-wise summation of o and linear offsets. The linear offsets are computed as follows:

$$\begin{aligned} \text{Input} &= \text{Concat}(\mathbf{x}_{-\text{p:t}}, \text{o}) \\ \text{Linear offsets} &= \tanh(\text{FC}(\text{Input})) \end{aligned} \tag{5.3}$$

the input of the FC layer is created by concatenating the recent observations and the initial predictions, which makes the FC layer aware of both the recent magnitude and the predicted waveform, reducing the difficulty of generating the linear offsets. The final predictions $\hat{\mathbf{x}}_{t+1:t+w}$ are obtained by summing the two terms:

$$\hat{\mathbf{x}}_{t+1:t+w} = o + \text{Linear offsets} \tag{5.4}$$

5.4 VEN

Unlike the small model size of TDANet, VEN is a much heavier model because it is built based on FC layers enabling a fast training process. To obtain accurate predictions while reducing the training time is not an easy task; even though RNN-based models can make good predictions, training these models is usually time-consuming which means RNN is not a suitable backbone architecture for building VEN. Among the existing forecasting models, N-BEATS [83] is a special one as it obtains good performance using only FC layers, which allows a fast-training; even though its performance can not be as good as the state-of-the-art RNN-based models, it proves that a model can obtain good forecasting performance using only FC layers along with residual connections. Inspired by the characteristics of RNN, we propose VEN which does away with the recurrent model and is completely built on FC layers; to achieve fast training speed of the model, we purposely avoid using computationally expensive mechanisms such as multi-head attention and others. The proposed model has a similar design as N-BEATS but follows a different underlying mechanism: while N-BEATS performs ensemble-style forecasting, our model focuses on modelling the variability of time series. In particular, the proposed architecture allows the network to introduce a certain level of variability in the time

series, on three temporal levels: daily observations, weekly observations and recent observations; on each level, the model generates series with different degrees of variability allowing the model to better describe the temporal dynamics of the input; eventually, the values at the last time step of all generated series are concatenated and used to make predictions. In this way, the proposed model can make accurate mobile traffic predictions while saving a lot of training time.

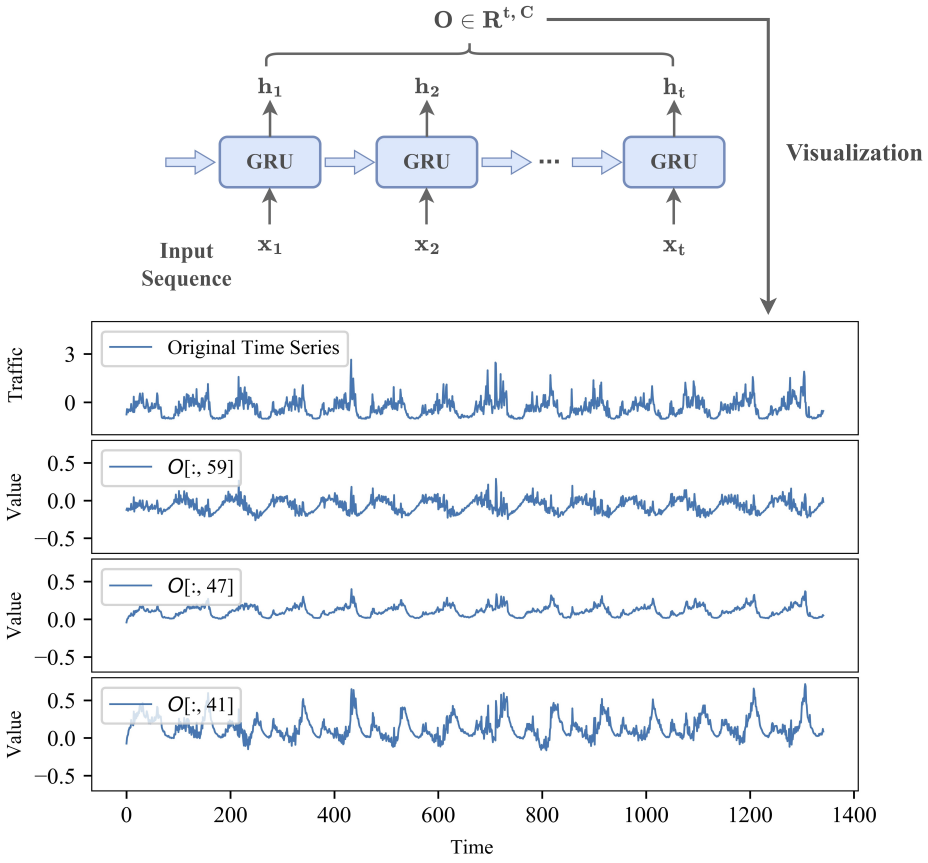


Fig. 5.3 Visualization of hidden vectors learned by GRU in time series forecasting task; O is the collection of hidden states whose shape is (t, C) where t is the number of input time steps and C is the number of channels of GRU.

5.4.1 Learning from RNN

While RNN-based models are known to be efficient at time series forecasting, their specific learning mechanism is not completely understood yet; Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are efficient at countering

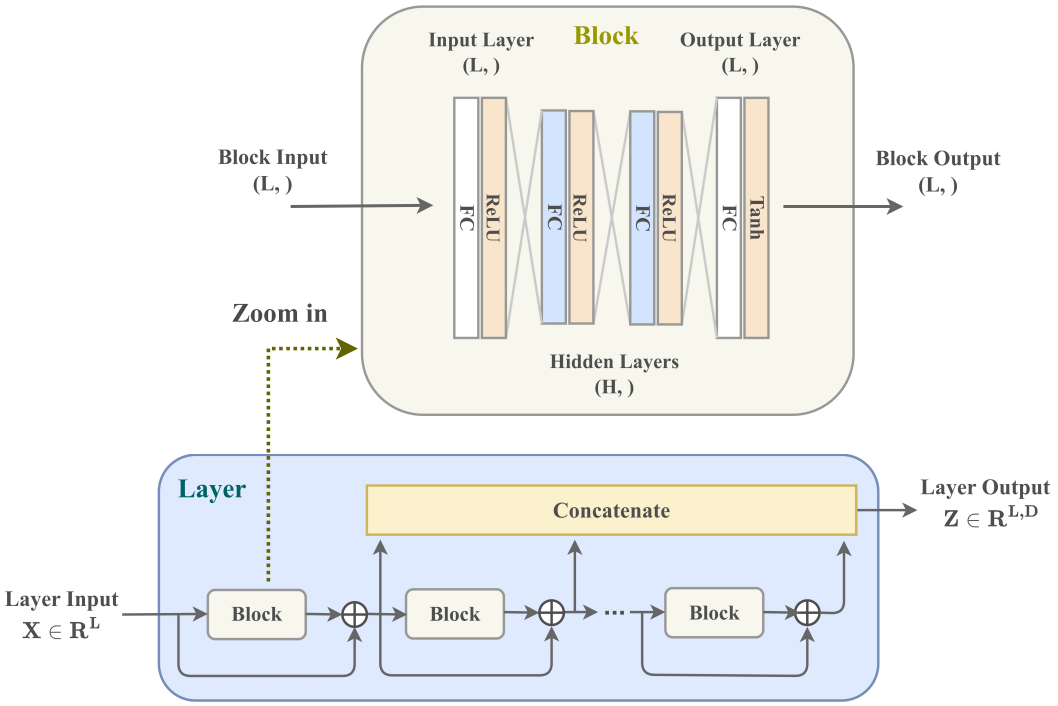


Fig. 5.4 Basic blocks and layers: a layer is composed of residual connected blocks whose depth is D (number of stacked blocks); each block will generate a new equal-length series which is used to enhance the variability, where L is the length of the block input and H is the number of hidden channels of the blue-coloured FC layers.

vanishing gradients, but the exact way they learn a model with temporal dependencies is unclear. To get an insight into RNNs characteristics, we have trained a GRU to predict the downlink usage two steps ahead, where the details of the dataset and experiment are discussed in Section 5.5. As shown in Figure 5.3, we visualize several extracted hidden vectors O, where each index (GRU channel) represents a specific hidden series learned from the input. If we compare the hidden series and the mobile traffic time series, we can find some similarities between them: hidden series still retains a temporal behaviour similar to that of the input series, although they have been processed by GRU leading to different levels of smoothing and noise. By observing the figure, it seems that GRU learns specific ways of scaling and introducing noises to the input series; the hidden features created by applying multiple levels of variability provide a better representation of the input space. Following this idea, a non RNN-based architecture may be as good as an RNN if it can learn how to modify the temporal dynamics of series gradually, while

the training time could be greatly reduced using only feedforward components; this is the inspiration for this work.

5.4.2 Basic building blocks

To present the design of VEN, we first introduce its basic building blocks and layers which are shown in Figure 5.4. The common way between our model and N-BEATS is that they both use residual connections and block-layer architecture, but their mechanisms are completely different: N-BEATS runs a sequential analysis of the input signal recursively, every time it forecasts part of the predictions and removes the well-modelled part from the previous signal, which performs an ensemble style forecasting. For our model, instead of performing ensemble forecasting, the layers are designed to model the unobserved hidden states of the last observations to better describe the distribution of the predicted random variable.

The backbone architecture of VEN consists of layers obtained by stacking a number of basic blocks, where each block consists of four FC layers. Blocks are used to generate a bias sequence whose length L is the same as the length of its input, and the blocks are connected in a residual way within the layers; every time a block creates a bias sequence, the generated series is added to the block input to introduce a certain degree of oscillation, eventually creating a variability-enhanced version of the input series. In the design of the block, we choose Rectified Linear Unit (ReLU) as the activation function for the first three FC layers to ensure the gradients can fast propagate across the block, and the last activation function is defined as the hyperbolic tangent function (Tanh) as we need to scale the value range of output to the range of -1 and 1 to ensure that every time only a small modification will be made to the block input. By stacking the blocks into a layer, it is feasible to gradually modify the layer input to get sequences with different degrees of variability; all the variability-enhanced series are collected and concatenated at the end which forms the output of the layers represented as $Z \in R^{L,D}$, where L is the input length and D is the number of stacked blocks (also known as the depth of the layer). Comparing GRU and the proposed layer, they employ different underlying mechanisms: a GRU will generate 32 hidden series if it has 32 channels, and all hidden series are generated at the same time at each time step by updating the previous hidden state. Conversely, at each time instant, a VEN layer generates a whole new hidden series without any recurring component.

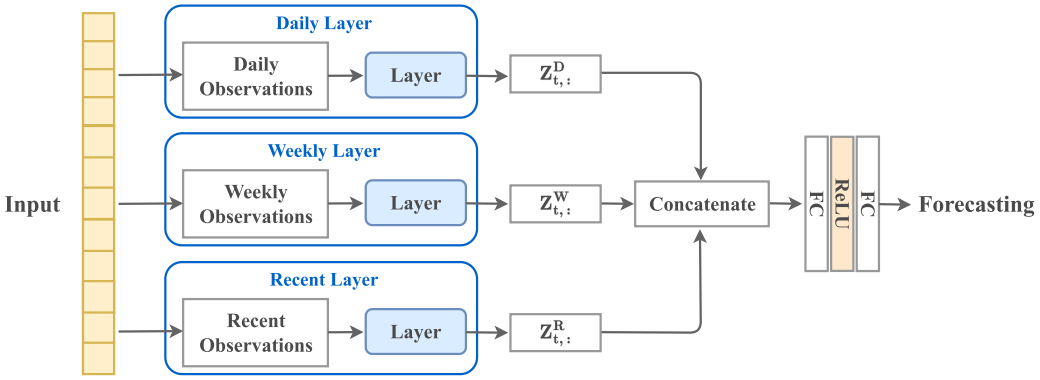


Fig. 5.5 Architecture of VEN, where $Z^D_{t,:}$, $Z^W_{t,:}$ and $Z^R_{t,:}$ are the values of the corresponding layer outputs at the last time step t.

5.4.3 Application to mobile traffic prediction

Figure 5.5 illustrates the architecture of VEN which is composed of three layers and two FC layers. Instead of using a single layer handling the whole input sequence, VEN uses three layers in parallel to process different time slots of the input series. The three layers are the daily layer, weekly layer and recent layer which handle daily observations, weekly observations and recent observations respectively. Generally speaking, in time series forecasting the useful information is relatively sparse which means most of the data points do not really help but rather add undesired noise to the output. In this case, processing the whole time series may not improve the forecasting performance but indeed significantly increase the processing time; here, we manually design three time windows to enhance the ability of VEN to make a trade-off among several temporal dependencies. Mobile traffic time series has complex temporal dynamics whose pattern is seen as a mixture of different terms; there are two important periodic patterns in mobile traffic including the daily pattern and the weekly pattern, where the daily pattern is mainly affected by human's daily schedule and the weekly pattern is dependent on the different between working day and weekend. Aside from these periodic patterns, the recent trend of mobile traffic is also very important considering that recent observations can be more useful to detect upcoming changes in mobile demand. Following the same idea presented in TDANet, for periodic patterns we only use the samples located at a local window, i.e., the samples just before and after the target instants $t' = t - nT$ with $n = 0, 1, \ldots,$ and T is referred to as the "skip period". The daily layer and weekly layer use the same local window setting but have different skip periods T; in the daily layer, the

skip period is set to 96 time steps which is the number of samples in one day, and the period is increased to 672 time steps (one week) in the weekly layer. In the recent layer, we use the observations in the last 24 hours to capture the recent changes. Once the model receives the outputs of all layers, only the values at the last time step are concatenated and used by FC layers to make predictions; the idea behind this is: the layer output can be seen as a collection of layer input series with different degrees of variability, where the last time step values represent the potential unobserved states of the last observation; the values encode the potential data distribution at the last observed time step allowing the model to learn a mapping between a distribution and predictions, this makes the model robust to the potential noise and achieve better generalization.

5.5 Experiments and results

5.5.1 Dataset

We conduct experiments on the dataset provided by Telecom Italia; the dataset includes the downlink usage data collected from the LTE network of a metropolitan city in Italy, describing the downlink mobile demand of 100 cells; three frequency layers are controlled by these cells, including 28 2600MHz cells, 43 1800MHz cells and 29 800MHz cells. This dataset consists of 32300 downlink usage time series, each time series is recorded during 14 consecutive days, and the traffic profile of each cell is aggregated over 15-minute intervals; our target is to predict the downlink demand for the next two steps (half an hour) and the next four steps (one hour). Notice that the time series of the dataset have been normalized with the standard score normalization and the normalized time series is calculated by first subtracting the mean value of the raw time series and then dividing by the standard deviation. The whole dataset has been split into the train set, validation set and test set with the ratios 80%, 10% and 10% respectively.

5.5.2 Benchmarks and performance metrics

In the experiments, we have implemented three predictors: TDANet, VEN and a single-layer predictor (it is composed of the basic layer of VEN and two FC layers).

To compare the forecasting performance of the proposed model, twelve baseline approaches are employed, including MLP [64], TCN [72], Transformer [88], DLinear [82], NLinear [82], N-BEATS [83], LSTM [78], GRU [79], DeepAR [86], MQ-RNN [87], LSTNet [84] and TPA-LSTM [85]. The baseline models cover the most popular approaches in DL-based time series forecasting, and they can be divided into two categories: non-RNN-based models (MLP, TCN, Transformer, DLinear, NLinear, N-BEATS) and RNN-based models (GRU, LSTM, DeepAR, MQ-RNN, LSTNet, TPA-LSTM); among these architectures, TPA-LSTM is one of the state-of-the-art forecasting models built based on RNN and temporal attention mechanisms. The models are trained to minimize the MAE on train set, and the hyperparameters have been tuned through grid search based on the performance evaluated on validation set. Once the best configurations have been determined, the models have been retrained on the dataset employed, by merging the train set and validation set and eventually evaluated on the test set. To evaluate the prediction performance, two metrics are used, namely MAE and MSE; besides MAE and MSE, #Parameters and training time are also used to evaluate the models, where #Parameters is used to evaluate how many parameters are included in the models, which is a good proxy of the complexity of a neural network, and training time is employed to compare the training speed of the methods. In order to compare the performance of the models, for each metric we use the bold text to indicate the best one and the underlined text to indicate the second-best one.

5.5.3 Results

We evaluate the performance of these models on test set, and the results are reported in Table 5.1. In this section, we present the results of TDANet and VEN separately.

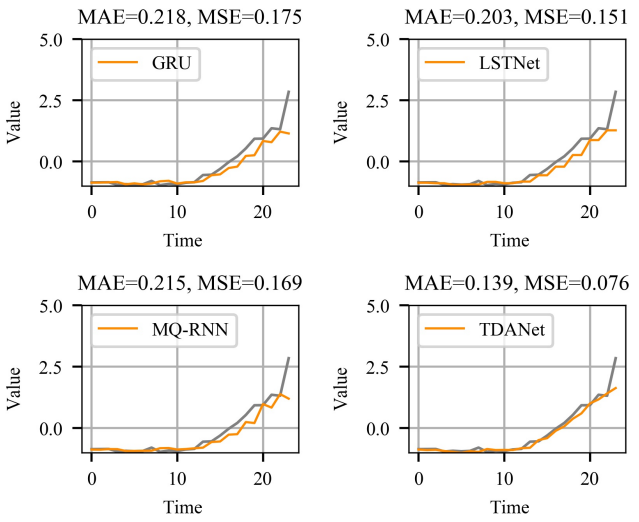
TDANet

Among the baseline approaches, we observe that the RNN-based models such as GRU, MQ-RNN and LSTNet obtain better performance compared to the others. Comparing to the baseline models, TDANet shows its superiority in mobile traffic prediction and outperforms all of them on all metrics for both 2 steps forecasting and 4 steps forecasting; TAP-LSTM is the only baseline method which can be as good as TDANet; however, it employs more parameters to model time series and the training

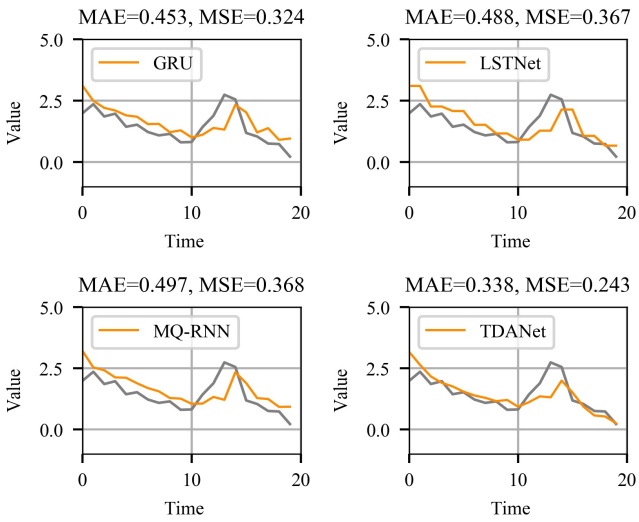
		w=2 (up to 30 minutes)			
	Model	MAE	MSE	#Parameters	Training Time
Non-RNN-Based	MLP	0.301	0.297	188k	**0.1 minutes**
	TCN	0.298	0.326	61k	24.7 minutes
	Transformer	0.332	0.342	1.12M	57.8 minutes
	DLinear	0.294	0.290	5k	0.3 minutes
	NLinear	0.298	0.293	**3k**	0.3 minutes
	N-BEATS	0.298	0.306	471k	0.9 minutes
RNN-Based	LSTM	0.295	0.303	265k	20.0 minutes
	GRU	0.286	0.298	50k	53.4 minutes
	DeepAR	0.314	0.313	265k	11.7 minutes
	MQ-RNN	0.290	0.299	141k	28.4 minutes
	LSTNet	0.282	0.287	102k	47.6 minutes
	TPA-LSTM	**0.278**	0.284	69k	218.0 minutes
Our	TDANet	**0.278**	0.284	33k	12.3 minutes
	VEN (depth=8)	0.279	**0.281**	425k	1.3 minutes

		w=4 (up to 60 minutes)			
	Model	MAE	MSE	#Parameters	Training Time
Non-RNN-Based	MLP	0.316	0.312	188k	**0.2 minutes**
	TCN	0.403	0.416	64k	27.2 minutes
	Transformer	0.382	0.390	1.12M	24.6 minutes
	DLinear	0.316	0.308	11k	0.3 minutes
	NLinear	0.319	0.306	**5k**	0.4 minutes
	N-BEATS	0.323	0.321	471k	1.2 minutes
RNN-Based	LSTM	0.326	0.331	265k	11.4 minutes
	GRU	0.313	0.330	50k	23.0 minutes
	DeepAR	0.358	0.359	265k	11.6 minutes
	MQ-RNN	0.319	0.320	174k	22.9 minutes
	LSTNet	0.302	0.307	102k	128.7 minutes
	TPA-LSTM	0.296	0.305	115k	138.6 minutes
Our	TDANet	**0.295**	**0.298**	36k	12.3 minutes
	VEN (depth=8)	0.298	0.301	549k	1.5 minutes

Table 5.1 Comparison of the performance of models: k and M are the shorthand notation for Thousand and Million; the bold text and underlined text are used to indicate the best and the second-best models, respectively.



(a)



(b)

Fig. 5.6 Comparison of the predictions made by RNN-based models. The gray line is the ground truth and the orange line is the forecasting; the y-axis is the normalized value of downlink usage. (a) increasing trend of mobile traffic; (b) decreasing trend of mobile traffic.

time is more than ten times slower than TDANet. From this point of view, TPA-LSTM is not as efficient as TDANet. As we presented previously, TDANet is a very lightweight model as it only has around 33k parameters; its model size is rather small considering its excellent forecasting performance. DLinear and NLinear are

	w=2 (up to 30 minutes)			
Model	MAE	MSE	#Parameters	Training Time
LSTM	0.295	0.303	<u>265k</u>	20.0 minutes
GRU	**0.286**	0.298	**50k**	53.4 minutes
Single-Layer Predictor (depth=8)	<u>0.287</u>	0.293	303k	0.9 minutes
Single-Layer Predictor (depth=16)	0.289	**0.290**	605k	**0.6 minutes**
Single-Layer Predictor (depth=32)	<u>0.287</u>	<u>0.291</u>	581k	<u>0.7 minutes</u>
	w=4 (up to 60 minutes)			
Model	MAE	MSE	#Parameters	Training Time
LSTM	0.326	0.331	265k	11.4 minutes
GRU	0.313	0.330	**50k**	23.0 minutes
Single-Layer Predictor (depth=8)	<u>0.306</u>	<u>0.305</u>	145k	<u>0.5 minutes</u>
Single-Layer Predictor (depth=16)	0.307	0.306	<u>131k</u>	**0.1 minutes**
Single-Layer Predictor (depth=32)	**0.304**	**0.303**	580k	1.1 minutes

Table 5.2 Comparison of RNNs and single layer predictors: k is the shorthand notation for Thousand; the bold text and underlined text are used to indicate the best and the second-best models, respectively.

lighter than TDANet considering that they are only composed of several FC layers, and its simple design results in worse forecasting performance compared to RNN-based models; in this case, TDANet is a good compromise between model size and forecasting performance. Another comparison can be made between TDANet and GRU: even though TDANet consists of two GRU components, its architecture can extract important features more efficiently, which means the GRU components do not require many parameters to learn the correlation of time series, eventually leading to even fewer parameters than baseline GRU model. From this point of view, the design of TDANet allows to model and predict mobile traffic in a more effective way with much fewer parameters, and the small model size makes it feasible to deploy TDANet on mobile or embedded devices whose memory is quite limited; hence, it is possible to use TDANet at the edge of a cellular network system to satisfy the specific mobile traffic forecasting needs of some applications, and further improve the quality of experience for end users.

As we discussed in the previous sections, the RNN-based models are not sensitive to the recent trend of time series, which is a drawback limiting their forecasting performance. In Figure 5.6, we visualize the forecasting of four RNN-based models; two scenarios are considered: increasing and decreasing trend of mobile traffic. In

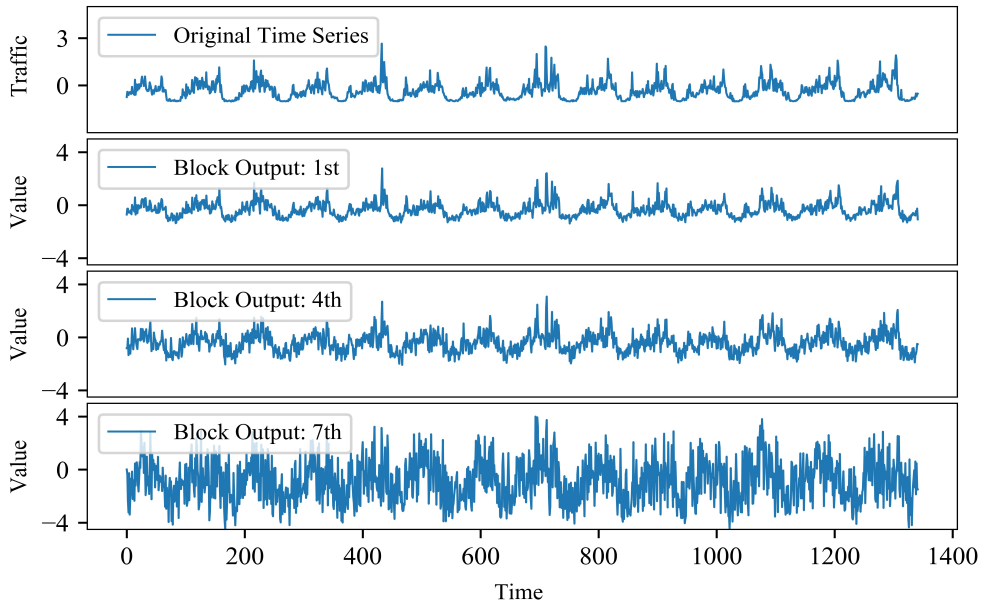


Fig. 5.7 Visualization of the variability-enhanced series generated by the single-layer predictor (depth=8).

the figure, it is obvious that the RNN-based baseline models cannot adapt very well; on the other hand, the proposed model can better follow the fast changes in the time series which leads to much lower MAE and MSE.

VEN

To better understand the behaviour of the proposed architecture, we first analyze the characteristics of the single-layer predictor; we have trained the single-layer predictors with three different depths and compare their performance with the two most widely used RNNs (LSTM and GRU). Table 5.2 evaluates the performance of RNNs and single-layer predictors, and the forecasting performance of single-layer models is better than RNNs from a global point of view: their predictions are as accurate as GRU when we make predictions two steps in advance, and their performance is significantly better than both GRU and LSTM if the forecasting horizon is 4; at the same time, single-layer models require much less training time and the price to pay is the larger model size. Comparing the performance of the predictors with different depths, we do not obtain obvious performance improvement by including more blocks; for this reason, the depth of the layers in VEN is set

to 8. Figure 5.7 visualizes the variability-enhanced series created by the blocks of single-layer predictor (depth=8), the sequences generated at three different depths are illustrated. If we observe the series, the output of the first block is very similar to the original time series and only a little perturbation is added to the input sequence; the introduced perturbation gradually increases as more blocks are used to add oscillation, eventually resulting in a very noisy series (the output of the seventh block). In this case, we can say that the degree of introduced variability is proportional to the depth of the layer. Based on these results, we prove that the behaviour of the basic layer follows our desired way, which obtains good forecasting performance by introducing different levels of variability to the input sequence while having a quite fast training speed compared to RNNs.

After discussing the characteristics and effectiveness of our proposed basic layer, we compare the performance of VEN with baseline approaches where results are reported in Table 5.1. According to the results, the best two baseline approaches are both RNN-based which are LSTNet and TPA-LSTM; the two models combine RNN architectures with other mechanisms such as the temporal attention mechanism, which improves the forecasting performance significantly. However, these 1 mechanisms are usually computationally slow and this makes LSTNet and TPA-LSTM have longer training time; besides this problem, LSTNet and TPA-LSTM are still seen as two state-of-the-art short-term forecasting models. For the non-RNN-based models, we can find that some models have very short training time; for example, the training time of MLP, NLinear and DLinear is less than half a minute; but at the same time, their forecasting performance is not that good. Even though Transformer and TCN are not built on RNNs, their training time is much longer considering that they either use a multi-head attention mechanism or stack many dilated convolutional layers. Compared to the best baseline model TPA-LSTM, VEN has almost the same performance; if we check the table, we can find that the difference in forecasting performance between VEN and TPA-LSTM is minor, but the training time of VEN is much shorter than TPA-LSTM: VEN requires approximately 1.5 minutes for training and the required training time of TPA-LSTM is hundreds of minutes. In this case, VEN obtains excellent forecasting performance while using only 1% training time of the best baseline approach. Even though N-BEATS is also a very fast model, its forecasting performance is much worse compared to RNN-based models and ours. Generally speaking, the forecasting performance of TDANet and VEN is similar meanwhile these models have different focuses: TDANet has a smaller model size

but longer training time and VEN has a larger model size but shorter training time. According to different mobile service scenarios, we can choose the most suitable one from them.

5.6 Conclusions

This chapter introduces two DL-based forecasting models: TDANet and VEN. The models are designed to address an important but usually neglected problem: how to predict mobile traffic cost-effectively under different mobile service scenarios; in addressing this problem, the main constraints are related to the large model size and long training time, which introduces additional difficulties in deploying forecasting models in the real world. To address these issues, TDANet and VEN are designed carefully to be a lightweight model and a fast-training model respectively, providing mobile operators with better choices of forecasting models under different scenarios.

Chapter 6

Conclusions

This thesis presents the works that have been done in the mobile traffic modelling field; by modelling and predicting mobile traffic more efficiently, mobile operators can better operate mobile networks and build intelligent telecommunication systems. In this chapter, we discuss the results we obtained, and outline the potential future challenges we will face.

6.1 The potential of AI-empowered mobile traffic modelling

In this thesis, we focused on developing algorithms built on ML and DL techniques for efficiently modelling mobile network KPIs. For mobile operators, how to configure mobile networks to satisfy the needs of mobile users is always a crucial problem for providing good mobile services. However, nowadays mobile networks have become rather complicated, leading to the high complexity of managing them; managing mobile networks automatically applying AI-based approaches is becoming the trend of today. Among the operational problems of mobile networks, two of the key topics are to extract useful knowledge from mobile data and obtain accurate mobile traffic predictions, which are essential for decision-making. To address these issues, different approaches have been proposed to meet the requirements of industrial applications under different real-world scenarios.

A critical problem encountered in mobile network management is the lack of prior knowledge with finer granularity. In complicated urban environments, the behaviour of base stations is difficult to understand as the usage patterns of mobile traffic can be both area-dependent and time-dependent. In this case, it is hard to allocate network resources to base stations because of the inhomogeneous distributed nature of user demand; if we want to optimize the resource allocation of mobile networks, it is beneficial to allocate network resources based on the importance level of each cell. During the monitoring of mobile networks, a massive amount of data has been recorded by mobile service providers, and it is a great source for building the knowledge base of cell importance. For identifying meaningful temporal patterns from mobile traffic, we develop a novel time series clustering algorithm in Chapter 3 namely TDC. In our theory, the cell importance is strongly related to the potential activeness of mobile users assigned to that cell, meaning that the core is to distinguish different degrees of the variability of mobile traffic time series. Based on this idea, TDC can create clusters concerning the variability of mobile traffic, providing a straightforward way to evaluate the cell importance. We conduct experiments using real-world mobile network datasets and TDC obtains excellent results on the case study of cell importance analysis; different regions are thought of as having very high user activeness such as stadiums, train stations and universities.

Besides extracting knowledge from recorded data, another intensively explored topic in this thesis is mobile traffic forecasting. Mobile traffic forecasting can be subdivided into two subjects: general mobile traffic forecasting and mobile traffic peak forecasting, where the latter has been rarely studied in the literature. To improve the QoS, the configuration of base stations has to be updated frequently to adapt to the recent mobile demand. In real-world applications, using the latest observations to guide the configuration update is thought an inefficient solution because of the existing observation delay; from this point of view, making accurate short-term mobile traffic forecasting plays a key role in configuration optimization. To improve the general mobile traffic forecasting performance, Chapter 5 presents two DL-based forecasting models which can predict mobile traffic accurately for the next hour. Moreover, the proposed predictors are even better as their design takes into account the deployment constraints met in industrial applications, which are often neglected. Among these approaches, TDANet is a lightweight model enabling the deployment on memory-limited devices or computational-extensive services, meanwhile VEN supports the fast-training making it easily used on large datasets when there is a

training time constraint. In general, improving mobile traffic peak forecasting is more critical compared to the general case because the peak is one of the most important temporal features of mobile traffic, and it usually refers to a social event or an unexpected burst of mobile traffic; a failure of peak prediction would result in a decrement in QoS and also affect the reliability of mobile services. However, mobile traffic peaks are extremely difficult to predict as they are rarely found in datasets and most of them occur suddenly for no apparent reason. To better predict mobile traffic peaks, we present two efficient predictors in Chapter 4 namely MoQ and FMLP. MoQ is a model built on the MoE framework which fuses the predictions of multiple experts. In order to overcome the problem of overly smooth predictions on peaks, the experts are designed to have differentiated forecasting styles from conservative to aggressive. A cooperation mechanism is established through a carefully designed training process, whereby conservative experts are responsible for the forecasting of the off-peak region, and the employed experts are switched to the aggressive ones once the potential increasing trend is detected by the manager, which leads to significantly improved peak predictions. Considering that MoQ is a "heavy" model using many parameters, FMLP is a lighter version which can predict mobile traffic peaks with fewer parameters while having worse forecasting performance compared to MoQ. According to the results, both of them have improved the peak forecasting performance by at least 50% compared to the widely used baseline approaches. Furthermore, the peak predictions are also employed by a predictive algorithm to determine if mobile traffic is distributed in an imbalanced way among neighbouring cells, and the results can be used to guide the decision-making of traffic handover between base stations.

6.2 Future challenges

In this thesis, we present different approaches regarding the topics of knowledge extraction and mobile traffic modelling. As mobile traffic has been increasing dramatically in recent days, we believe that there is still a lot of work having to be done in the aforementioned fields. In Chapter 4, even though the proposed approaches can make good peak predictions, there is still a gap between the overall forecasting performance of peak forecasting models and the state-of-the-art general forecasting models; how to design a model retaining the advantages of both of them

would be an important topic. By considering TDC (Chapter 3) can create groups based on the variability of time series, it could be interesting to study which cells play key roles in network optimization by integrating TDC with other methods. Due to the fact that the forecasting difficulty of each cluster is different, it is possible to design forecasting models individually for each cluster to better learn the temporal pattern of time series, which would potentially improve the performance of mobile traffic forecasting. Once mobile operators employ well-designed modelling solutions, we believe that the maintenance difficulty would be reduced a lot and the management would be highly efficient, though it will take a significant amount of effort.

References

[1] Martino Trevisan, Danilo Giordano, Idilio Drago, Marco Mellia, and Maurizio Munafo. Five years at the edge: Watching internet from the isp network. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*, pages 1–12, 2018.

[2] Ericsson. Ericsson annual report 2022. 2023.

[3] Mamta Agiwal, Abhishek Roy, and Navrati Saxena. Next generation 5g wireless networks: A comprehensive survey. *IEEE communications surveys & tutorials*, 18(3):1617–1655, 2016.

[4] Akhil Gupta and Rakesh Kumar Jha. A survey of 5g network: Architecture and emerging technologies. *IEEE access*, 3:1206–1232, 2015.

[5] Kan Zheng, Zhe Yang, Kuan Zhang, Periklis Chatzimisios, Kan Yang, and Wei Xiang. Big data-driven optimization for mobile networks toward 5g. *IEEE network*, 30(1):44–51, 2016.

[6] Chunxiao Jiang, Haijun Zhang, Yong Ren, Zhu Han, Kwang-Cheng Chen, and Lajos Hanzo. Machine learning paradigms for next-generation wireless networks. *IEEE Wireless Communications*, 24(2):98–105, 2016.

[7] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. Deep learning in mobile and wireless networking: A survey. *IEEE Communications surveys & tutorials*, 21(3):2224–2287, 2019.

[8] Fengli Xu, Yong Li, Huandong Wang, Pengyu Zhang, and Depeng Jin. Understanding mobile traffic patterns of large scale cellular towers in urban environment. *IEEE/ACM transactions on networking*, 25(2):1147–1161, 2016.

[9] Ermias Andargie Walelgne, Alemnew Sheferaw Asrese, Jukka Manner, Vaibhav Bajpai, and Jörg Ott. Clustering and predicting the data usage patterns of geographically diverse mobile users. *Computer Networks*, 187:107737, 2021.

[10] Angelo Furno, Marco Fiore, and Razvan Stanica. Joint spatial and temporal classification of mobile traffic demands. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[11] Cristina Marquez, Marco Gramaglia, Marco Fiore, Albert Banchs, Cezary Ziemlicki, and Zbigniew Smoreda. Not all apps are created equal: Analysis of spatiotemporal heterogeneity in nationwide mobile service usage. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*, pages 180–186, 2017.

[12] Coşkun Hamzaçebi, Diyar Akay, and Fevzi Kutay. Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting. *Expert systems with applications*, 36(2):3839–3844, 2009.

[13] Oscar Claveria and Salvador Torra. Forecasting tourism demand to catalonia: Neural networks vs. time series models. *Economic Modelling*, 36:220–228, 2014.

[14] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, pages 1394–1401. IEEE, 2018.

[15] Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern recognition letters*, 42:11–24, 2014.

[16] Xi Ouyang, Chaoyun Zhang, Pan Zhou, Hao Jiang, and Shimin Gong. Deepspace: An online deep learning framework for mobile big data to understand human mobility patterns. *arXiv preprint arXiv:1610.07009*, 2016.

[17] Cheng Yang, Maosong Sun, Wayne Xin Zhao, Zhiyuan Liu, and Edward Y Chang. A neural network approach to jointly modeling social networks and mobile trajectories. *ACM Transactions on Information Systems (TOIS)*, 35(4):1–28, 2017.

[18] Qingsong Feng, Yabin Zhang, Chao Li, Zheng Dou, and Jin Wang. Anomaly detection of spectrum in wireless communication via deep auto-encoders. *The Journal of Supercomputing*, 73:3161–3178, 2017.

[19] Donghwoon Kwon, Hyunjoo Kim, Jinoh Kim, Sang C Suh, Ikkyun Kim, and Kuinam J Kim. A survey of deep learning-based network anomaly detection. *Cluster Computing*, 22:949–961, 2019.

[20] Mahmood Yousefi-Azar, Vijay Varadharajan, Len Hamey, and Uday Tupakula. Autoencoder-based feature learning for cyber security applications. In *2017 International joint conference on neural networks (IJCNN)*, pages 3854–3861. IEEE, 2017.

[21] Imad Alawe, Adlen Ksentini, Yassine Hadjadj-Aoul, and Philippe Bertin. Improving traffic forecasting for 5g core network scalability: A machine learning approach. *IEEE Network*, 32(6):42–49, 2018.

[22] Jie Feng, Xinlei Chen, Rundong Gao, Ming Zeng, and Yong Li. Deeptp: An end-to-end neural network for mobile cellular traffic prediction. *IEEE Network*, 32(6):108–115, 2018.

[23] Chih-Wei Huang, Chiu-Ti Chiang, and Qiuhui 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)*, pages 1–6. IEEE, 2017.

[24] Mehdi Roopaei, Paul Rad, and Mo Jamshidi. Deep learning control for complex and large scale cloud systems. *Intelligent Automation & Soft Computing*, 23(3):389–391, 2017.

[25] Michele Zorzi, Andrea Zanella, Alberto Testolin, Michele De Filippo De Grazia, and Marco Zorzi. Cobanets: A new paradigm for cognitive communications systems. In *2016 International Conference on Computing, Networking and Communications (ICNC)*, pages 1–7. IEEE, 2016.

[26] Bo Ma, Weisi Guo, and Jie Zhang. A survey of online data-driven proactive 5g network optimisation using machine learning. *IEEE access*, 8:35606–35637, 2020.

[27] Po-Chiang Lin, Lionel F Gonzalez Casanova, Bakary KS Fatty, et al. Data-driven handover optimization in next generation mobile communication networks. *Mobile Information Systems*, 2016, 2016.

[28] Kashif Sultan, Hazrat Ali, and Zhongshan Zhang. Call detail records driven anomaly detection and traffic prediction in mobile cellular networks. *IEEE Access*, 6:41728–41737, 2018.

[29] Dooyoung Kim, Bongjhin Shin, Daehyoung Hong, and Jaechan Lim. Self-configuration of neighbor cell list utilizing e-utran nodeb scanning in lte systems. In *2010 7th IEEE Consumer Communications and Networking Conference*, pages 1–5. IEEE, 2010.

[30] Seyed Morteza Mousavi, Hamid Reza Rabiee, M Moshref, and A Dabirmoghaddam. Mobility pattern recognition in mobile ad-hoc networks. In *Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology*, pages 302–309, 2007.

[31] Qinghe Du, Houbing Song, Qian Xu, Pinyi Ren, and Li Sun. Interference-controlled d2d routing aided by knowledge extraction at cellular infrastructure towards ubiquitous cps. *Personal and ubiquitous Computing*, 19:1033–1043, 2015.

[32] Jianpeng Li, Kun Zhu, and Yang Zhang. Knowledge-assisted few-shot fault diagnosis in cellular networks. In *2022 IEEE Globecom Workshops (GC Wkshps)*, pages 1292–1297. IEEE, 2022.

[33] Linyu Huang, Liang Lu, and Wei Hua. A survey on next-cell prediction in cellular networks: Schemes and applications. *IEEE Access*, 8:201468–201485, 2020.

[34] Javier Villegas, Eduardo Baena, Sergio Fortes, and Raquel Barco. Social-aware forecasting for cellular networks metrics. *IEEE Communications Letters*, 25(6):1931–1934, 2021.

[35] Sergio Fortes, David Palacios, Inmaculada Serrano, and Raquel Barco. Applying social event data for the management of cellular networks. *IEEE Communications Magazine*, 56(11):36–43, 2018.

[36] Weiting Zhang, Dong Yang, and Hongchao Wang. Data-driven methods for predictive maintenance of industrial equipment: A survey. *IEEE systems journal*, 13(3):2213–2227, 2019.

[37] Tiago Zonta, Cristiano André Da Costa, Rodrigo da Rosa Righi, Miromar Jose de Lima, Eduardo Silveira da Trindade, and Guann Pyng Li. Predictive maintenance in the industry 4.0: A systematic literature review. *Computers & Industrial Engineering*, 150:106889, 2020.

[38] Prapanna Mondal, Labani Shit, and Saptarsi Goswami. Study of effectiveness of time series modeling (arima) in forecasting stock prices. *International Journal of Computer Science, Engineering and Applications*, 4(2):13, 2014.

[39] Alfred Cowles. Stock market forecasting. *Econometrica, Journal of the Econometric Society*, pages 206–214, 1944.

[40] Manfred Mudelsee. Trend analysis of climate time series: A review of methods. *Earth-science reviews*, 190:310–322, 2019.

[41] Gian Antonio Susto, Angelo Cenedese, and Matteo Terzi. Time-series classification methods: Review and applications to power systems data. *Big data application in power systems*, pages 179–220, 2018.

[42] Sai Manoj Pudukotai Dinakarrao, Axel Jantsch, and Muhammad Shafique. Computer-aided arrhythmia diagnosis with bio-signal processing: A survey of trends and techniques. *ACM Computing Surveys (CSUR)*, 52(2):1–37, 2019.

[43] Zhenyu Liu, Zhengtong Zhu, Jing Gao, and Cheng Xu. Forecast methods for time series data: a survey. *Ieee Access*, 9:91896–91912, 2021.

[44] George EP Box and David A Pierce. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American statistical Association*, 65(332):1509–1526, 1970.

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

[46] Everette S Gardner Jr. Exponential smoothing: The state of the art. *Journal of forecasting*, 4(1):1–28, 1985.

[47] Peter R Winters. Forecasting sales by exponentially weighted moving averages. *Management science*, 6(3):324–342, 1960.

[48] Samuel Asuamah Yeboah, Manu Ohene, TB Wereko, et al. Forecasting aggregate and disaggregate energy consumption using arima models: a literature survey. *Journal of Statistical and Econometric Methods*, 1(2):71–79, 2012.

[49] Siu Lau Ho and Min Xie. The use of arima models for reliability forecasting and analysis. *Computers & industrial engineering*, 35(1-2):213–216, 1998.

[50] Evangelos D Spyrou, Ioannis Tsoulos, and Chrysostomos Stylios. Applying and comparing lstm and arima to predict co levels for a time-series measurements in a port area. *Signals*, 3(2):235–248, 2022.

[51] Vaia I Kontopoulou, Athanasios D Panagopoulos, Ioannis Kakkos, and George K Matsopoulos. A review of arima vs. machine learning approaches for time series forecasting in data driven networks. *Future Internet*, 15(8):255, 2023.

[52] Ayodele Ariyo Adebiyi, Aderemi Oluyinka Adewumi, Charles Korede Ayo, et al. Comparison of arima and artificial neural networks models for stock price prediction. *Journal of Applied Mathematics*, 2014, 2014.

[53] Eranga De Saa and Lochandaka Ranathunga. Comparison between arima and deep learning models for temperature forecasting. *arXiv preprint arXiv:2011.04452*, 2020.

[54] Eva Ostertagova and Oskar Ostertag. Forecasting using simple exponential smoothing method. *Acta Electrotechnica et Informatica*, 12(3):62, 2012.

[55] Gianluca Bontempi, Souhaib Ben Taieb, and Yann-Aël Le Borgne. Machine learning strategies for time series forecasting. *Business Intelligence: Second European Summer School, eBISS 2012, Brussels, Belgium, July 15-21, 2012, Tutorial Lectures 2*, pages 62–77, 2013.

[56] Pedro Lara-Benítez, Manuel Carranza-García, and José C Riquelme. An experimental review on deep learning architectures for time series forecasting. *International journal of neural systems*, 31(03):2130001, 2021.

[57] Alan Lapedes and Robert Farber. Nonlinear signal processing using neural networks: Prediction and system modelling. Technical report, 1987.

[58] Paul Werbos. Beyond regression: New tools for prediction and analysis in the behavioral sciences. *PhD thesis, Committee on Applied Mathematics, Harvard University, Cambridge, MA*, 1974.

[59] Paul J Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356, 1988.

[60] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

[61] Lin Wang, Zhigang Wang, Hui Qu, and Shan Liu. Optimal forecast combination based on neural networks for time series forecasting. *Applied soft computing*, 66:1–17, 2018.

[62] Huaizhi Wang, Zhenxing Lei, Xian Zhang, Bin Zhou, and Jianchun Peng. A review of deep learning for renewable energy forecasting. *Energy Conversion and Management*, 198:111799, 2019.

[63] Benedikt Pfülb, Christoph Hardegen, Alexander Gepperth, and Sebastian Rieger. A study of deep learning for network traffic data forecasting. In *Artificial Neural Networks and Machine Learning–ICANN 2019: Text and Time Series: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings, Part IV 28*, pages 497–512. Springer, 2019.

[64] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[65] Alfonso Palmer, Juan Jose Montano, and Albert Sesé. Designing an artificial neural network for forecasting tourism time series. *Tourism management*, 27(5):781–790, 2006.

[66] G Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

[67] Nikolaos Kourentzes, Devon K Barrow, and Sven F Crone. Neural network ensemble operators for time series forecasting. *Expert Systems with Applications*, 41(9):4235–4244, 2014.

[68] Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Forecasting stock prices from the limit order book using convolutional neural networks. In *2017 IEEE 19th conference on business informatics (CBI)*, volume 1, pages 7–12. IEEE, 2017.

[69] Ping-Huan Kuo and Chiou-Jye Huang. A high precision artificial neural networks model for short-term energy load forecasting. *Energies*, 11(1):213, 2018.

[70] Irena Koprinska, Dengsong Wu, and Zheng Wang. Convolutional neural networks for energy time series forecasting. In *2018 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2018.

[71] Mengmeng Cai, Manisa Pipattanasomporn, and Saifur Rahman. Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques. *Applied energy*, 236:1078–1088, 2019.

[72] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[73] Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. Dilated convolutional neural networks for time series forecasting. *Journal of Computational Finance, Forthcoming*, 2018.

[74] Renzhuo Wan, Shuping Mei, Jun Wang, Min Liu, and Fan Yang. Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting. *Electronics*, 8(8):876, 2019.

[75] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[76] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802–808, 2018.

[77] Yuhuang Hu, Adrian Huber, Jithendar Anumula, and Shih-Chii Liu. Overcoming the vanishing gradient problem in plain recurrent networks. *arXiv preprint arXiv:1801.06105*, 2018.

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

[79] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[80] Benjamin Lindemann, Timo Müller, Hannes Vietz, Nasser Jazdi, and Michael Weyrich. A survey on long short-term memory networks for time series prediction. *Procedia CIRP*, 99:650–655, 2021.

[81] Sameh Mahjoub, Larbi Chrifi-Alaoui, Bruno Marhic, and Laurent Delahoche. Predicting energy consumption using lstm, multi-layer gru and drop-gru neural networks. *Sensors*, 22(11):4062, 2022.

[82] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? *arXiv preprint arXiv:2205.13504*, 2022.

[83] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

[84] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.

[85] Shun-Yao Shih, Fan-Keng Sun, and Hung-yi Lee. Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, 108:1421–1441, 2019.

[86] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

[87] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.

[88] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[89] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[90] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430, 2021.

[91] Hansika Hewamalage, Klaus Ackermann, and Christoph Bergmeir. Forecast evaluation for data scientists: common pitfalls and best practices. *Data Mining and Knowledge Discovery*, 37(2):788–832, 2023.

[92] Glenn W Milligan and Martha C Cooper. Methodology review: Clustering methods. *Applied psychological measurement*, 11(4):329–354, 1987.

[93] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. Time-series clustering–a decade review. *Information systems*, 53:16–38, 2015.

[94] Eamonn Keogh and Jessica Lin. Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowledge and information systems*, 8:154–177, 2005.

[95] Ahmet Mert and Aydin Akan. Hilbert-huang transform based hierarchical clustering for eeg denoising. In *21st European Signal Processing Conference (EUSIPCO 2013)*, pages 1–5. IEEE, 2013.

[96] Aristides Gionis and Heikki Mannila. Finding recurrent sources in sequences. In *Proceedings of the seventh annual international conference on Research in computational molecular biology*, pages 123–130, 2003.

[97] Gautam Das, King-Ip Lin, Heikki Mannila, Gopal Renganathan, and Padhraic Smyth. Rule discovery from time series. In *KDD*, volume 98, pages 16–22. Citeseer, 1998.

[98] Sherri K Harms, Steve Goddard, Stephen E Reichenbach, William J Waltman, and Tsegaye Tadesse. Data mining in a geospatial decision support system for drought risk management. In *Proceedings of the 2001 National Conference on Digital Government Research*, pages 9–16, 2002.

[99] Sherri K Harms, Jitender Deogun, and Tsegaye Tadesse. Discovering sequential association rules with constraints and time lags in multiple sequences. In *International symposium on methodologies for intelligent systems*, pages 432–441. Springer, 2002.

[100] John Paparrizos and Luis Gravano. k-shape: Efficient and accurate clustering of time series. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 1855–1870, 2015.

[101] L.G.B. Ruiz, M.C. Pegalajar, R. Arcucci, and M. Molina-Solana. A time-series clustering methodology for knowledge extraction in energy consumption data. *Expert Systems with Applications*, 160:113731, 2020.

[102] Félix Iglesias and Wolfgang Kastner. Analysis of similarity measures in times series clustering for the discovery of building energy patterns. *Energies*, 6(2):579–597, 2013.

[103] Ville Hautamaki, Pekka Nykanen, and Pasi Franti. Time-series clustering by approximate prototypes. In *2008 19th International conference on pattern recognition*, pages 1–4. IEEE, 2008.

[104] T Warren Liao. Clustering of time series data—a survey. *Pattern recognition*, 38(11):1857–1874, 2005.

[105] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[106] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009.

[107] Raymond T. Ng and Jiawei Han. Clarans: A method for clustering objects for spatial data mining. *IEEE transactions on knowledge and data engineering*, 14(5):1003–1016, 2002.

[108] Chonghui Guo, Hongfeng Jia, and Na Zhang. Time series clustering based on ica for stock data analysis. In *2008 4th international conference on wireless communications, networking and mobile computing*, pages 1–4. IEEE, 2008.

[109] Jessica Lin, Michail Vlachos, Eamonn Keogh, and Dimitrios Gunopulos. Iterative incremental clustering of time series. In *Advances in Database Technology-EDBT 2004: 9th International Conference on Extending Database Technology, Heraklion, Crete, Greece, March 14-18, 2004 9*, pages 106–122. Springer, 2004.

[110] Joseph C Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. 1973.

[111] Xiaozhe Wang, Kate Smith, and Rob Hyndman. Characteristic-based clustering for time series data. *Data mining and knowledge Discovery*, 13:335–364, 2006.

[112] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[113] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[114] Xingyu Cai, Tingyang Xu, Jinfeng Yi, Junzhou Huang, and Sanguthevar Rajasekaran. DTWNet: a Dynamic Time Warping Network. *Advances in Neural Information Processing Systems*, 32, 2019.

[115] Marco Cuturi and Mathieu Blondel. Soft-DTW: a differentiable loss function for time-series. In *International Conference on Machine Learning*, pages 894–903. PMLR, 2017.

[116] Weiwei Jiang. Cellular traffic prediction with machine learning: A survey. *Expert Systems with Applications*, 201:117163, 2022.

[117] Hyun-Woo Kim, Jun-Hui Lee, Yong-Hoon Choi, Young-Uk Chung, and Hyukjoon Lee. Dynamic bandwidth provisioning using arima-based traffic forecasting for mobile wimax. *Computer Communications*, 34(1):99–106, 2011.

[118] Li-Na Wang, Chen-Rui Zang, and Yuan-Yuan Cheng. The short-term prediction of the mobile communication traffic based on the product seasonal model. *SN Applied Sciences*, 2:1–9, 2020.

[119] Quang Thanh Tran, Li Hao, and Quang Khai Trinh. A comprehensive research on exponential smoothing methods in modeling and forecasting cellular traffic. *Concurrency and Computation: Practice and Experience*, 32(23):e5602, 2020.

[120] Diogo Clemente, Gabriela Soares, Daniel Fernandes, Rodrigo Cortesao, Pedro Sebastiao, and Lucio S. Ferreira. Traffic forecast in mobile networks: Classification system using machine learning. In *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, pages 1–5, 2019.

[121] Jin Huang and Ming Xiao. Mobile network traffic prediction based on seasonal adjacent windows sampling and conditional probability estimation. *IEEE Transactions on Big Data*, 8(5):1155–1168, 2020.

[122] Anil Kirmaz, Diomidis S Michalopoulos, Irina Balan, and Wolfgang Gerstacker. Mobile network traffic forecasting using artificial neural networks. In *2020 28th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 1–7. IEEE, 2020.

[123] Basma Mahdy, Hazem Abbas, Hossam S Hassanein, Aboelmagd Noureldin, and Hatem Abou-zeid. A clustering-driven approach to predict the traffic load of mobile networks for the analysis of base stations deployment. *Journal of Sensor and Actuator Networks*, 9(4):53, 2020.

[124] Yuchao Zhu and Shaowei Wang. Joint traffic prediction and base station sleeping for energy saving in cellular networks. In *ICC 2021-IEEE International Conference on Communications*, pages 1–6. IEEE, 2021.

[125] S Swedha and ES Gopi. Lstm network for hotspot prediction in traffic density of cellular network. In *Machine Learning, Deep Learning and Computational Intelligence for Wireless Communication: Proceedings of MDCWC 2020*, pages 35–47. Springer, 2021.

[126] Varun Kurri, Vishweshvaran Raja, and P Prakasam. Cellular traffic prediction on blockchain-based mobile networks using lstm model in 4g lte network. *Peer-to-Peer Networking and Applications*, 14:1088–1105, 2021.

[127] Chuanting Zhang, Haixia Zhang, Dongfeng Yuan, and Minggao Zhang. Citywide cellular traffic prediction based on densely connected convolutional neural networks. *IEEE Communications Letters*, 22(8):1656–1659, 2018.

[128] Yin Gao, Man Zhang, Jiajun Chen, Jiren Han, Dapeng Li, and Ruitao Qiu. Accurate load prediction algorithms assisted with machine learning for network traffic. In *2021 International Wireless Communications and Mobile Computing (IWCMC)*, pages 1683–1688, 2021.

[129] Shanjun Zhan, Lisu Yu, Zhen Wang, Yichen Du, Yan Yu, Qinghua Cao, Shuping Dang, and Zahid Khan. Cell traffic prediction based on convolutional neural network for software-defined ultra-dense visible light communication networks. *Security and Communication Networks*, 2021:1–10, 2021.

[130] Zekai Chen, Jiaze E, Xiao Zhang, Hao Sheng, and Xiuzheng Cheng. Multitask time series forecasting with shared attention. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pages 917–925, 2020.

[131] Qingyao Liu, Jianwu Li, and Zhaoming Lu. St-tran: Spatial-temporal transformer for cellular traffic prediction. *IEEE Communications Letters*, 25(10):3325–3329, 2021.

[132] Dehai Zhang, Linan Liu, Cheng Xie, Bing Yang, and Qing Liu. Citywide cellular traffic prediction based on a hybrid spatiotemporal network. *Algorithms*, 13(1):20, 2020.

[133] Qingtian Zeng, Qiang Sun, Geng Chen, and Hua Duan. Attention based multicomponent spatiotemporal cross-domain neural network model for wireless cellular network traffic prediction. *EURASIP Journal on Advances in Signal Processing*, 2021:1–25, 2021.

[134] Wenxin Shen, Haixia Zhang, Shuaishuai Guo, and Chuanting Zhang. Timewise attention aided convolutional neural network for data-driven cellular traffic prediction. *IEEE Wireless Communications Letters*, 10(8):1747–1751, 2021.

[135] Feng Zheng and Quanyun Liu. Anomalous telecom customer behavior detection and clustering analysis based on ISP's operating data. *IEEE Access*, 8:42734–42748, 2020.

[136] Ermias Andargie Walelgne, Alemnew Sheferaw Asrese, Jukka Manner, Vaibhav Bajpai, and Jörg Ott. Clustering and predicting the data usage patterns of geographically diverse mobile users. *Computer Networks*, 187:107737, 2021.

[137] Kashif Sultan, Hazrat Ali, and Zhongshan Zhang. Call detail records driven anomaly detection and traffic prediction in mobile cellular networks. *IEEE Access*, 6:41728–41737, 2018.

[138] Qiqi Zhu and Li Sun. Big data driven anomaly detection for cellular networks. *IEEE Access*, 8:31398–31408, 2020.

[139] Hoang Duy Trinh, Engin Zeydan, Lorenza Giupponi, and Paolo Dini. Detecting mobile traffic anomalies through physical control channel fingerprinting: A deep semi-supervised approach. *IEEE Access*, 7:152187–152201, 2019.

[140] Fengli Xu, Yong Li, Huandong Wang, Pengyu Zhang, and Depeng Jin. Understanding mobile traffic patterns of large scale cellular towers in urban environment. *IEEE/ACM Transactions on Networking*, 25(2):1147–1161, 2017.

[141] Shuyang Li, Gianluca Francini, and Enrico Magli. Temporal dynamics clustering for analyzing cell behavior in mobile networks. *Computer Networks*, 223:109578, 2023.

[142] Alexander J McNeil. Extreme value theory for risk managers. *Departement Mathematik ETH Zentrum*, 12(5):121–237, 1999.

[143] Carl Scarrott and Anna MacDonald. A review of extreme value threshold estimation and uncertainty quantification. *REVSTAT–Statistical Journal*, 10(1):33–60, 2012.

[144] Mico Loretan and Peter C.B. Phillips. Testing the covariance stationarity of heavy-tailed time series: An overview of the theory with applications to several financial datasets. *Journal of Empirical Finance*, 1(2):211–248, 1994.

[145] Ada K. F. Ho and Alan T. K. Wan. Testing for covariance stationarity of stock returns in the presence of structural breaks: an intervention analysis. *Applied Economics Letters*, 9(7):441–447, 2002.

[146] MF Omran and E McKenzie. Testing for covariance stationarity in the uk all-equity returns. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 48(3):361–369, 1999.

[147] Marie Kratz and Sidney I. Resnick. The qq-estimator and heavy tails. *Communications in Statistics. Stochastic Models*, 12(4):699–724, 1996.

[148] Holger Drees, Sidney Resnick, and Laurens de Haan. How to make a Hill plot. *The Annals of Statistics*, 28(1):254 – 274, 2000.

[149] Cláudia Neves and M.I. Fraga Alves. Reiss and Thomas' automatic selection of the number of extremes. *Computational Statistics and Data Analysis*, 47(4):689–704, 2004.

[150] Erich Schubert and Peter J Rousseeuw. Faster k-medoids clustering: improving the PAM, CLARA, and CLARANS algorithms. In *International Conference on Similarity Search and Applications*, pages 171–187. Springer, 2019.

[151] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[152] David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979.

[153] T. Caliński and J Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27, 1974.

[154] Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar, and Eli Woods. Tslearn, a machine learning toolkit for time series data. *Journal of Machine Learning Research*, 21(118):1–6, 2020.

[155] Andrei Novikov. PyClustering: Data mining library. *Journal of Open Source Software*, 4(36):1230, apr 2019.

[156] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[157] Xuan Liang, Tao Zou, Bin Guo, Shuo Li, Haozhe Zhang, Shuyi Zhang, Hui Huang, and Song Xi Chen. Assessing beijing's PM2.5 pollution: severity, weather impact, apec and winter heating. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2182):20150257, 2015.

[158] Shuyang Li, Enrico Magli, Gianluca Francini, and Giorgio Ghinamo. Deep learning based prediction of traffic peaks in mobile networks. *Computer Networks*, page 110167, 2023.

[159] Shuyang Li, Enrico Magli, and Gianluca Francini. To be conservative or to be aggressive? a risk-adaptive mixture of experts for mobile traffic forecasting. In *ICC 2023-IEEE International Conference on Communications*, pages 5471–5476. IEEE, 2023.

[160] Ahmad Awada, Bernhard Wegmann, Ingo Viering, and Anja Klein. A joint optimization of antenna parameters in a cellular network using taguchi's method. In *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE, 2011.

[161] Ahmad Awada, Bernhard Wegmann, Ingo Viering, and Anja Klein. A mathematical model for user traffic in coverage and capacity optimization of a cellular network. In *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE, 2011.

[162] Henrik Klessig, Albrecht Fehske, Gerhard Fettweis, and Jens Voigt. Improving coverage and load conditions through joint adaptation of antenna tilts and cell selection rules in mobile networks. In *2012 International Symposium on Wireless Communication Systems (ISWCS)*, pages 21–25. IEEE, 2012.

[163] Raymond Kwan, Rob Arnott, Robert Paterson, Riccardo Trivisonno, and Mitsuhiro Kubota. On mobility load balancing for lte systems. In *2010 IEEE 72nd vehicular technology conference-fall*, pages 1–5. IEEE, 2010.

[164] Shaoshuai Fan, Hui Tian, and Cigdem Sengul. Self-optimization of coverage and capacity based on a fuzzy neural network with cooperative reinforcement learning. *EURASIP Journal on Wireless Communications and Networking*, 2014(1):1–14, 2014.

[165] Víctor Buenestado, Matías Toril, Salvador Luna-Ramírez, José María Ruiz-Avilés, and Adriano Mendo. Self-tuning of remote electrical tilts based on call traces for coverage and capacity optimization in lte. *IEEE Transactions on Vehicular Technology*, 66(5):4315–4326, 2016.

[166] Ying Yang, Pengfei Li, Xiaohui Chen, and Weidong Wang. A high-efficient algorithm of mobile load balancing in lte system. In *2012 IEEE Vehicular Technology Conference (VTC Fall)*, pages 1–5. IEEE, 2012.

[167] Hoang Duy Trinh, Lorenza Giupponi, and Paolo 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)*, pages 1827–1832. IEEE, 2018.

[168] Chaoyun Zhang and Paul Patras. Long-term mobile traffic forecasting using deep spatio-temporal neural networks. In *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 231–240, 2018.

[169] Nikolay Dandanov, Hussein Al-Shatri, Anja Klein, and Vladimir Poulkov. Dynamic self-optimization of the antenna tilt for best trade-off between coverage and capacity in mobile networks. *Wireless Personal Communications*, 92(1):251–278, 2017.

[170] Rouzbeh Razavi, Siegfried Klein, and Holger Claussen. Self-optimization of capacity and coverage in lte networks using a fuzzy reinforcement learning approach. In *21st Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1865–1870. IEEE, 2010.

[171] Tao Cai, Georgios P Koudouridis, Christer Qvarfordt, Johan Johansson, and Peter Legg. Coverage and capacity optimization in e-utran based on central coordination and distributed gibbs sampling. In *2010 IEEE 71st Vehicular Technology Conference*, pages 1–5. IEEE, 2010.

[172] Nikolay Dandanov, Soumya Ranjan Samal, Shuvabrata Bandopadhaya, Vladimir Poulkov, Krasimir Tonchev, and Pavlina Koleva. Comparison of wireless channels for antenna tilt based coverage and capacity optimization. In *2018 Global Wireless Summit (GWS)*, pages 119–123. IEEE, 2018.

[173] Cheng Yanyun, Huet Alexis, Xu Hui, and Yan Xingxiu. Coverage and capacity optimization for 4g lte networks using differential evolution. In *2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, pages 640–645. IEEE, 2018.

[174] Raymond Kwan, Rob Arnott, Robert Paterson, Riccardo Trivisonno, and Mitsuhiro Kubota. On mobility load balancing for lte systems. In *2010 IEEE 72nd Vehicular Technology Conference - Fall*, pages 1–5, 2010.

[175] Ridha Nasri and Zwi Altman. Handover adaptation for dynamic load balancing in 3gpp long term evolution systems. *arXiv preprint arXiv:1307.1212*, 2013.

[176] Ning Zhang, Shan Zhang, Shaohua Wu, Ju Ren, Jon W Mark, and Xuemin Shen. Beyond coexistence: Traffic steering in lte networks with unlicensed bands. *IEEE wireless communications*, 23(6):40–46, 2016.

[177] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[178] Ronan Collobert, Samy Bengio, and Yoshua Bengio. A parallel mixture of svms for very large scale problems. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002.

[179] Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

[180] Shlomo E Chazan, Jacob Goldberger, and Sharon Gannot. Speech enhancement using a deep mixture of experts. *arXiv preprint arXiv:1703.09302*, 2017.

[181] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[182] ZongYuan Ge, Christopher McCool, Conrad Sanderson, and Peter Corke. Subset feature learning for fine-grained category classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 46–52, 2015.

[183] Ravi Teja Mullapudi, William R Mark, Noam Shazeer, and Kayvon Fatahalian. Hydranets: Specialized dynamic architectures for efficient inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8080–8089, 2018.

[184] David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314*, 2013.

[185] Roger Koenker and Gilbert Bassett Jr. Regression quantiles. *Econometrica: journal of the Econometric Society*, pages 33–50, 1978.

[186] James W Taylor. A quantile regression neural network approach to estimating the conditional density of multiperiod returns. *Journal of Forecasting*, 19(4):299–311, 2000.

[187] Wenjie Zhang, Hao Quan, and Dipti Srinivasan. Parallel and reliable probabilistic load forecasting via quantile regression forest and quantile determination. *Energy*, 160:810–819, 2018.

[188] Filipe Rodrigues and Francisco C Pereira. Beyond expectation: Deep joint mean and quantile regression for spatiotemporal problems. *IEEE transactions on neural networks and learning systems*, 31(12):5377–5389, 2020.

[189] Shuyang Li, Enrico Magli, and Gianluca Francini. Tdanet: An efficient solution for short-term mobile traffic forecasting. In *2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*, pages 1–5. IEEE, 2023.

[190] Shuyang Li, Enrico Magli, and Gianluca Francini. Exploring time series variability: A training-efficient mobile traffic predictor. In *WCNC 2024-IEEE Wireless Communications and Networking Conference*, pages 1–6. IEEE, 2024.

[191] Jiansheng Lin, Youjia Chen, Haifeng Zheng, Ming Ding, Peng Cheng, and Lajos Hanzo. A data-driven base station sleeping strategy based on traffic prediction. *IEEE Transactions on Network Science and Engineering*, 2021.

[192] Abida Perveen, Raouf Abozariba, Mohammad Patwary, and Adel Aneiba. Dynamic traffic forecasting and fuzzy-based optimized admission control in federated 5g-open ran networks. *Neural Computing and Applications*, pages 1–19, 2021.

[193] Qing He, György Dán, and Georgios P Koudouridis. Semi-persistent scheduling for 5g downlink based on short-term traffic prediction. In *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pages 1–6. IEEE, 2020.

[194] Thilina N Weerasinghe, Indika AM Balapuwaduge, and Frank Y Li. Supervised learning based arrival prediction and dynamic preamble allocation for bursty traffic. In *IEEE INFOCOM 2019-IEEE conference on computer communications workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE, 2019.

[195] Zekai Chen, E Jiaze, Xiao Zhang, Hao Sheng, and Xiuzheng Cheng. Multi-task time series forecasting with shared attention. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pages 917–925. IEEE, 2020.