Deep Multi-task Learning Models for Short-term Supply-demand Forecasting in Ride-hailing System

by

Md. Hishamur Rahman

MASTER OF SCIENCE IN CIVIL ENGINEERING

November, 2020

Recommendation of the Board of Examiners

The thesis titled "Deep Multi-task Learning Models for Short-term Supply-demand Forecasting in Ride-hailing System" submitted by Md. Hishamur Rahman, Std. No. 125605 of Academic Year 2012-13 has been found as satisfactory and accepted as partial fulfillment of the requirement for the degree of Master of Science in Civil Engineering.

Dr. Shakil Mohammad Rifaat Professor Department of Civil and Environmental Engineering (CEE) Islamic University of Technology (IUT)	Chairman
Dr. Md. Rezaul Karim Professor & Head Department of Civil and Environmental Engineering (CEE) Islamic University of Technology (IUT)	Member (Ex-Officio)
Dr. Moinul Hossain Professor Department of Civil and Environmental Engineering (CEE) Islamic University of Technology (IUT)	Member
Dr. Nazmus Sakib Assistant Professor Department of Civil and Environmental Engineering (CEE) Islamic University of Technology (IUT)	Member
Dr. Md. Shahid Mamun Professor and Head, Department of Civil Engineering Ahsanullah University of Science and Technology (AUST)	Member (External)

Declaration of Candidate

It is hereby declared that this thesis/project report or any part of it has not been submitted elsewhere for the award of any Degree or Diploma.

Supervisor:
Dr. Shakil Mohammad Rifaat
Professor
Department of Civil and
Environmental Engineering
Islamic University of Technology
Date:

Candidate:
Md. Hishamur Rahman
Student No: 125605
Date:

Dedication

I like to dedicate this thesis to my father, my mother, and especially my wife and my daughter.

Table of Contents

Soft Copy (CD)

List of Tables

List of Figures

List of Abbreviations

TNC	Transportation Network Companies
POI	Point of Interest
GBM	Gradient Boosting Machine
GBDT	Gradient Boosting Decision Tree
XGBoost	Extreme Gradient Boosting
RF	Random Forest
SVM	Support Vector Machine
ANN	Artificial Neural Network
AR	Auto-regressive
MA	Moving Average
ARIMA	Auto-regressive Integrated Moving Average
GLM	Generalized Linear Model
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-term Memory
GRU	Gated Recurrent Units
GCN	Graph Convolutional Network
ConvRNN	Convolutional Recurrent Neural Network
ConvLSTM	Convolutional Long Short-term Memory
ME	Mixture of Experts
Conv-ME	Convolutional Mixture of Experts
GRU-ME	Gated Recurrent Unit with Mixture of Experts
ConvRNN-ME	Convolutional Recurrent Neural Network with Mixture of Experts

FOCIR-Net	Feature importance integrated One-dimensional Convolution and Recurrent Network
OCIR-Net	One-dimensional Convolution and Independently Recurrent Network
FOC-Net	Feature importance integrated One-dimensional Convolution Network
FIR-Net	Feature importance integrated Independently Recurrent Network
GESME-Net	Gated Ensemble of Spatio-temporal Mixture of Experts Network
SESME-Net	Single-gated Ensemble of Spatio-temporal Mixture of Experts Network
SBSM-Net	Shared Bottom Spatio-temporal Mixture Network
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
sMAPE	Symmetric Mean Absolute Percentage Error
LASSO	Least Absolute Shrinkage and Selection Operator

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor Prof. Dr. Shakil Mohammad Rifaat, for his valuable guidance, advice, and support. Without his help and encouragement, this work would never be complete. His open-mindedness helped me to choose my desired topic of research and conduct my research work independently. I also would like to express my sincere thanks to supervisory committee members Prof. Dr. Md. Shahid Mamun, Prof. Dr. Md. Moinul Hossain and Prof. Dr. Nazmus Sakib for their constructive comments on my thesis which greatly improved the quality of the work.

I want to express my gratefulness to Prof. Dr. Md. Rezaul Karim, Head, Department of Civil and Environmental Engineering, Islamic University of Technology for providing all sorts of support during the thesis.

I would like to thank Ehsan Karim, Head of Data Science, Pathao for informing me about the state-of-the-art practices and problems to focus on during my research. Special thanks to my colleague Soumik Nafees Sadeek and mentee Masnun Abrar for their assistance during the thesis.

A big thanks to Didi Chuxing for the publicly available datasets and the open datasets under the GAIA Open Data Initiative. Without their data, it would not have been possible to conduct this research.

I am forever indebted to my parents for continuously encouraging me in all my pursuits and my wife and daughter for sacrificing their family time to support my work till the end.

Abstract

Transportation network companies (TNCs) are providing on-demand door-to-door ride-hailing services in many cities around the world. In order to reduce passenger waiting time and driver search friction, TNCs need to conduct spatio-temporal forecasting of demand and supply-demand gap. However, due to spatio-temporal dependencies pertaining to demand and supply-demand gap in a ride-hailing system, making accurate forecasts for demand and supply-demand gap is a difficult task. Although spatio-temporal deep learning methods have recently proven to be successful in detecting the spatio-temporal dependencies, there exist few more challenges in implementing these methods that require further attention.

One of the assumptions in spatio-temporal online taxi-hailing demand forecasting with deep learning is that spatio-temporal dependencies rely on spatial structure and therefore, the zone-wise historical data is processed as image pixels. However, spatio-temporal dependencies among the zones are complex and do not capture patterns like image pixels. Furthermore, commonly applied two-dimensional convolution and long-short term memory (LSTM) to detect spatio-temporal patterns from zone-wise historical data increases model complexity, which is not justified with respect to a spatio-temporal deep learning baseline. Therefore, this study applies one-dimesional convolution to historical data with flattened zones to investigate the effectiveness of preserving spatial structure in spatio-temporal forecasting of on-demand taxis and develops a spatio-temporal deep learning baseline architecture containing one-dimensional convolutional recurrent layers for justifying increased model complexity in architectures containing two-dimensional convolutional recurrent layers. Experiments with real-world online taxi-hailing data of Didi Chuxing from Chengdu show that convolutional recurrent model implementing one-dimensional convolution in vanilla recurrent neural network with rectified linear activation outperforms convolutional recurrent models combining two-dimensional convolution and long short-term memory network. The findings indicate that preserving spatial structure in online taxi-hailing demand forecasting can be sometimes redundant and complex spatio-temporal deep learning models should be compared to a spatio-temporal deep learning baseline in order to build more computationally efficient architectures for spatio-temporal forecasting.

Due to confidentiality and privacy issues, ride-hailing data are sometimes released to the researchers by removing spatial adjacency information of the zones, which hinders the detection of spatio-temporal dependencies in deep learning. To that end, a novel spatio-temporal deep learning architecture is proposed in this study for forecasting demand and supply-demand gap in a ride-hailing system with anonymized spatial adjacency information, which integrates feature importance layer with a spatio-temporal deep learning architecture containing one-dimensional convolutional neural network (CNN) and zone-distributed independently recurrent neural network (IndRNN). The developed architecture is tested with real-world datasets of Didi Chuxing, which shows that models developed based on the proposed architecture can outperform machine learning models (e.g., gradient boosting machine, distributed random forest, generalized linear model, artificial neural network). Additionally, the feature importance layer provides an interpretation of the model by revealing the contribution of the input features utilized in prediction.

Designing and maintaining the spatio-temporal forecasting models separately in a task-wise and city-wise manner is a challenging task for the continuously expanding TNCs. therefore, a deep multi-task learning architecture is proposed in this study by developing a gated ensemble of spatio-temporal mixture of experts network (GESME-Net) containing convolutional recurrent neural network (CRNN), convolutional neural network (CNN), and recurrent neural network (RNN), which is capable of simultaneously forecasting different spatio-temporal tasks in a city as well as same task across different cities. The proposed architecture is tested with real-world data of Didi Chuxing for: (i) simultaneous forecasting of demand and supply-demand gap in Beijing, and (ii) simultaneous forecasting of demand for Chengdu and Xian. In both scenarios, models from the proposed architecture outperformed the multi-task learning benchmark (e.g., shared bottom model, single-gated spatio-temporal mixture of experts), task-wise and city-wise spatio-temporal deep learning models, and machine learning algorithms (e.g., gradient boosting, random forest, and generalized linear models). The developed architecture provides a basis for spatio-temporal multi-task learning in smart cities.

The developments made in this study bridges the gap between applying advanced deep learning methods and maintaining the privacy of the TNCs' data at the same time, provides a way for reducing the computational cost of spatio-temporal deep learning models for the

TNCs, and importantly, shows the viability of utilizing spatio-temporal multi-task learning architecture for jointly forecasting demand and supply-demand gap in a ride-hailing system.

CHAPTER 1: INTRODUCTION

1.1 Background

Online taxi-hailing/ride-hailing companies, also known as Transportation Network Companies (TNCs), are providing on-demand door-to-door transportation service. Ride-hailing is recognized as a disruptive urban transportation mode where registered private car owners provide on-demand rides by driving their vehicles [1]. With the advancement of mobile technologies, ride-hailing services are replacing conventional taxi services and reshaping the mode choice behavior of passengers. In the last few years, several ride-hailing companies such as Uber, Lyft, Didi Chuxing have gained increasing popularity in many cities around the world. A recent study found that around 20% of adults in an urban area are using ride-hailing services [2]. Ride-hailing services are conducted through mobile applications, which coordinate the matching of passengers' requests with vacant cars.

The TNCs operate through a mobile application and act as a trip making agent between the customer and the driver. As a result, they can collect data that include Global Positioning System (GPS) traces and trip details. With the increasing use of ride-hailing services, a large amount of data is being collected at an enormous scale. For example, Didi Chuxing, the popular TNC from China, has more than 550 million users worldwide, making more than 10 billion trips per year, resulting in more than 4875 terabytes of data every day to feed its predictive and optimization algorithms [3]. Likewise, other TNCs around the world are also collecting and utilizing such large-scale data to improve their service quality. Importantly, the question of how to utilize this massive data to improve spatio-temporal forecasting in ride-hailing system is receiving increasing attention in the transportation research community and therefore, it is chosen as the primary focus of this study.

In the spatio-temporal forecasting, the data of any time interval is aggregated in a matrix based on geometrically partitioned regions, which can then be processed as image pixels in deep learning models. From a spatial-zoning perspective, understanding and improving the spatio-temporal forecasting of short-term demand and supply-demand gap is immensely important to ride-hailing companies for determining the spatial allocation of fleet vehicles and reducing the spatial disequilibrium of supply-demand. However, due to the limited availability of ride-hailing data, less focus has been given to improve the spatio-temporal forecasting of demand and supply-demand gap in a ride-hailing system. Nevertheless, due to the similarity between taxi service and ride-hailing service, the concepts of supply-demand in the taxi system can shed important light on the ride-hailing system. Supply-demand equilibrium can be achieved theoretically if three parameters become equal: the rate of passengers demanding rides, the rate of available taxis, and the matching rate [4]. However, equilibrium cannot be maintained continuously due to information asymmetry [5]. Therefore, disequilibrium can occur from excessive supply or excessive demand, which is unfavourable for both taxi companies and ride-hailing companies, thus necessitating them to stress on the accurate forecasting of demand and supply-demand gap. On the one hand, forecasting the demand can assist the ride-hailing companies in finding hotspot demand zones and determining appropriate fleet size. While on the other hand, forecasting the supply-demand gap serves as an indicator of the supply deficit of ride-hailing cars in each zone, which can be informative for implementing dynamic pricing strategies and incentivizing drivers as a means to deal with the disequilibrium issues. Nevertheless, there exist challenges in both the forecasting tasks due to spatio-temporal dependencies [6], [7], which are not arising from the same factors. The relative difficulties in forecasting demand and supply-demand gap of a ride-hailing system can be addressed by two contrasting factors:

1) Local vs. non-local spatial dependencies: Ride-hailing supply-demand gap shows strong local spatial dependencies because of the continuous and rapid movement of vacant cars across nearby zones, while the passengers' zone of trip origin remains almost unchanged [8].

Therefore, the supply-demand gap of a zone can be fulfilled by the vacant cars of the surrounding zones, suggesting spatially correlated supply-demand gap patterns among the neighbouring zones. However, for ride-hailing demand, non-local spatial dependencies (e.g., functional similarity, transportation connectivity) are considered in addition to the local spatial dependencies [9], all of which cannot be specified without relying on data mining.

2) Strong vs. weak temporal dependencies: Ride-hailing demand is found to have strong temporal dependencies due to almost recurring travel patterns of passengers every day, whereas ride-hailing supply-demand gap has weak temporal dependencies because of irregular fluctuations in the supply of vacant ride-hailing cars [8].

The conventional approach for determining transportation forecasts is known as four-step modeling, which includes trip generation, trip distribution, mode choice, and traffic assignment [10]. An alternative technique to this ordered approach is developed by combining trip generation, distribution, and mode choice in a single equation, which is known as direct demand model [11], [12]. The direct demand modeling approach received some attention because of its advantage in overcoming the problem of step-by-step error accumulation in the four-step modeling approach [13]. However, the direct demand model has received limited attention in the intraurban context [14], due to unavailability of large amount of trip data [15] and inability to incorporate insights from individual travel behavior [16]. Recently, with the availability of large amount of trip data for public transit thanks to the technological advancement in intelligent transportation systems, there is a growing trend in utilizing direct demand models for transit demand forecasting [17], [18].

Recent studies showed that machine learning algorithms allow a more flexible model structure to capture the complex non-linear patterns among variables, which often outperforms conventional statistical models in terms of predictive accuracy for transportation forecasts [19], [20]. The recent success of machine learning algorithms provide a different

perspective to the direct demand modeling as a zone-wise time-series forecasting problem. However, due to the complex structure of the machine learning models, these models cannot explain the impacts of policy changes and only focuses on producing reliable short-term transportation forecasts. Following this track, this study concentrates on improving the predictive accuracy of zone-wise demand and supply-demand gap in a ride-hailing system through deep learning. Although a large number of studies already utilized deep learning to improve the predictive accuracy in spatio-temporal forecasting problems, few other associated problems require further attention that are covered in this study.

1.1.1 Deep Learning Model Complexity in Spatio-temporal Forecasting

With the increasing computational power, researchers are shifting from classical statistical methods to neural network-based methods for traffic forecasting problems [21]. In recent years, deep learning has been used widely by the researchers across different domains for various types of forecasting problems because of its ability of learning complex features by processing huge datasets and providing state-of-the-art prediction models [22]. Many researchers were inspired by this success and applied deep learning methods to various forecasting problems related to transportation systems, such as bus ridership forecasting [23], crowd flow forecasting [24], [25], human trajectories forecasting [26], [27], online taxi-hailing demand forecasting [4], crash prediction [28], crash severity forecasting [29], traffic congestion forecasting [30], [31], traffic flow forecasting [32]–[36], traffic speed forecasting [37]–[39], and travel time forecasting [40], [41].

In the spatio-temporal demand forecasting, the data of any time interval is aggregated in a matrix based on geometrically partitioned regions, which can then be processed as image pixels in deep learning models. One of the main challenges in spatio-temporal demand forecasting with deep learning is how to deal with spatial dependencies. To model the spatial correlations, convolutional neural network (CNN) [42] is found to be an effective algorithm

to learn the spatial correlations. Another challenge for spatio-temporal demand forecasting lies in the detection of the temporal dependencies. The recurrent neural network (RNN) [43] is found as one of the most suitable algorithms for learning the temporal correlations. One exclusive property of RNN architecture is that the variables in one time-slot are correlated with the variables in the previous time-slots [44]. However, the conventional RNN suffers from vanishing gradient problem that limits the storing of long-term information [45]. To overcome this limitation, the long short-term memory (LSTM) network [45] employs a series of memory cells for storing long-term information, which is found to be effective for learning long-term temporal dependencies.

Recent studies on demand forecasting and traffic forecasting have successfully implemented convolutional recurrent layers for jointly modeling the spatial and temporal dependencies [8], [46]. However, spatio-temporal correlations in these studies are modeled by preserving the spatial structure in the historical data, processing data as image pixels and applying two-dimensional convolution to detect the spatial correlations, which may lead to unnecessarily increased model complexity if it is not justified with respect to one-dimensional convolution. Furthermore, to prevent vanishing/exploding gradient problems in convolutional recurrent layers, convolutional LSTM [47] has been developed. However, the use of LSTM in the convolutional recurrent layer makes it computationally expensive due to a large number of output units [48] and does not provide better model performance other than that they only ease the training process [49]. Rather, a convolutional recurrent layer with a vanilla RNN can also tackle the vanishing/exploding gradient problem by utilizing a linear activation function, while being computationally cheaper. In this study, one-dimensional convolution is utilized on historical data with flattened zones to investigate the effectiveness of preserving spatial structure in spatio-temporal forecasting of on-demand taxis and develop a spatio-temporal deep learning baseline containing one-dimensional convolutional recurrent layers.

1.1.2 Anonymized Spatial Adjacency Information in Spatio-temporal Forecasting

The most popular methods to detect spatio-temporal dependencies are the convolutional neural network (CNN) [42] and the recurrent neural network (RNN) [43], which achieved outstanding success in tasks related to computer vision and natural language processing. Earlier studies that applied deep learning for spatio-temporal forecasting generally divide the whole study area into several zones, and the historical features corresponding to the zones are utilized as inputs in the CNN or RNN. However, these methods require further considerations for spatio-temporal forecasting with ride-hailing data containing anonymized spatial adjacency information of the zones. Firstly, previous studies on spatio-temporal forecasting with deep learning require known spatial adjacency information for processing through two-dimensional CNN, which hinders the use of two-dimensional CNN in the case of data with anonymized spatial adjacency information. However, it is possible to process input features of such data as one-dimensional signals through one-dimensional CNN, based on recent findings that showed the applicability of CNN to learn from scrambled images [50]. Secondly, although gradient vanishing/exploding problem in the RNN for capturing long-term temporal dependencies can be controlled by utilizing the independently recurrent neural network (IndRNN) [51], applying such method directly to learn temporal dependencies from spatio-temporal features is not possible due to the internal structure of the RNN.

In this study, a spatio-temporal deep learning architecture, the feature importance integrated one-dimensional convolution and independently recurrent network (FOCIR-Net), is put forward to deal with the spatio-temporal dependencies for forecasting demand and supply-demand gap in a ride-hailing system with anonymized spatial adjacency information. The feature importance layer utilized in the architecture, in addition to providing an interpretation of the model, emphasizes more on strong predictors and attenuates weak predictors for each zone. To the best of the author's knowledge, this study, for the first time, integrates feature importance with a spatio-temporal deep learning architecture and applies one-dimensional

CNN and zone-distributed IndRNN to deal with spatio-temporal dependencies in ride-hailing data with anonymized spatial adjacency information, which can be utilized to forecast both demand and supply-demand gap in a ride-hailing system.

1.1.3 Problem of Spatio-temporal Multi-task Learning

To ensure high-quality service with real-time information and timely mobility of the passengers, the ride-hailing companies need to forecast different indicators (e.g., demand and supply-demand gap) at the same time in a city as well as across the cities they are operating. Addressing these forecasting tasks individually require separate design, maintenance, and updating of forecasting architectures, which is a computational burden for the ride-hailing companies. Therefore, the question arises — can we model multiple forecasting tasks in a ride-hailing system with a unified architecture to ease that burden?

Spatio-temporal forecasting has been widely studied in the last decade to enhance the efficiency of ride-hailing companies by incorporating cutting-edge computational tools. In ride-hailing system, the two most important issues of spatio-temporal forecasting are demand and supply-demand gap. However, there exist challenges in both forecasting tasks due to variations in spatio-temporal dependencies [6], [7], which follow converse directions.

Interestingly, spatio-temporal dependencies for forecasting demand and supply-demand gap in a city are modeled using the same type of features, yet separate architectures are utilized for their prediction without considering a common representation from these features. Furthermore, the usual approach of these forecasting architecture is to utilize information that are related to the city of interest only. The same spatio-temporal forecasting task (e.g., demand) in different cities are modeled using the corresponding dataset of the city without considering the correlation among the features of the cities that can provide better inductive bias. Therefore, a multi-task learning framework that has the ability to capture these

correlations in a city as well as across the cities with a joint representation can substantially reduce the computational burden for the ride-hailing companies. However, for simultaneously forecasting multiple spatio-temporal tasks in a ride-hailing system along with capturing spatio-temporal dependencies in a city as well as across cities, spatio-temporal deep learning methods require modification in their forecasting architectures to incorporate multi-task learning, which is not yet done till now.

Inspired by the success of deep learning methods for modeling spatio-temporal forecasting problems, an overlooked aspect is explored in this study—modeling multiple spatio-temporal forecasting tasks of a city or across cities in a ride-hailing system by a multi-task learning architecture. While such multi-task learning architectures have been used in natural language processing [52], machine translation [53], speech recognition [54], and computer vision problems [55], but never been applied in spatio-temporal forecasting problems in ride-hailing system. Previously, for spatio-temporal forecasting problems in ride-hailing system, deep learning was applied to deal only with the problem at hand, which limits the efficiency of deep learning since repeating efforts are required for each problem [56]. Motivated by the works of J. Ma et al. [57], a spatio-temporal multi-task learning architecture with mixture-of-experts will be developed in this study for forecasting multiple spatio-temporal tasks in a city as well as across cities.

1.2 Objective of the Study

The main objectives of this study are:

- To investigate the effectiveness of preserving spatial structure in spatio-temporal forecasting of on-demand taxis and develop a spatio-temporal deep learning baseline containing one-dimensional convolutional recurrent layers.

- To develop a spatio-temporal deep learning architecture in order to deal with the spatio-temporal dependencies for forecasting demand and supply-demand gap in a ride-hailing system with anonymized spatial adjacency information.
- To develop a spatio-temporal multi-task learning architecture with mixture-of-experts for forecasting multiple spatio-temporal tasks in a city as well as across cities.

1.3 Scope of Research

In this study, one-dimensional convolution is utilized on historical data with flattened zones to investigate the effectiveness of preserving spatial structure in spatio-temporal forecasting of on-demand taxis and develop a spatio-temporal deep learning baseline containing one-dimensional convolutional recurrent layers. The proposed deep learning architecture is evaluated with a real-world online taxi-hailing dataset of Didi from Chengdu.

Furthermore, a spatio-temporal deep learning architecture, the feature importance integrated one-dimensional convolution and independently recurrent network (FOCIR-Net), is put forward in this study to deal with the spatio-temporal dependencies for forecasting demand and supply-demand gap in a ride-hailing system with anonymized spatial adjacency information. The feature importance layer utilized in the architecture, in addition to providing an interpretation of the model, emphasises more on strong predictors and attenuates weak predictors for each zone. The proposed deep learning architecture is evaluated with two real-world ride-hailing datasets of Didi from Beijing and Hangzhou.

A spatio-temporal multi-task learning architecture with mixture-of-experts is finally developed in this study for forecasting multiple spatio-temporal tasks in a city as well as across cities. The proposed multi-task learning architecture is tested with real-world datasets of Didi for two scenarios of multi-task learning in ride-hailing system: (i) simultaneous

forecasting of demand and supply-demand gap in Beijing, and (ii) simultaneous forecasting of demand for Chengdu and Xian.

1.4 Significance of the Research

The developments made in this study bridges the gap between applying advanced deep learning methods and maintaining the confidentiality of the TNCs' data at the same time. Furthermore, the developments provide a way for reducing the computational cost of spatio-temporal deep learning models for the TNCs, while improving the accuracy for spatio-temporal forecasting of ride-hailing demand and supply-demand gap in a ride-hailing system. Last but not least, the multi-task learning architecture developed in this study shows the viability of jointly forecasting demand and supply-demand gap in a ride-hailing system with a single model without compromising the accuracy. Technically, this study will contribute to the research literature in the following ways:

1. A one-dimensional convolutional recurrent neural network (1D-ConvRNN) is developed by implementing one-dimensional convolution in a vanilla recurrent neural network with rectified linear activation, which can be utilized as a spatio-temporal deep learning baseline for justifying increased model complexity in spatio-temporal forecasting with deep learning.

2. The effectiveness of preserving spatial structure in spatio-temporal forecasting is investigated by applying one-dimensional convolution on historical data with flattened zones and comparing them with two-dimensional convolution applied on historical data processed as images, which is not done before.

3. The FOCIR-Net, for the first time, detects spatial dependencies from the anonymized zones in a multivariate manner through one-dimensional CNN and temporal dependencies in an independent manner through zone-distributed IndRNN.

4. This study, for the first time, integrates feature importance with a spatio-temporal deep learning architecture and applies one-dimensional CNN and zone-distributed IndRNN to deal with spatio-temporal dependencies in ride-hailing data with anonymized spatial adjacency information, which can be utilized to forecast both demand and supply-demand gap in a ride-hailing system. The feature importance layer integrated with a spatio-temporal deep learning architecture determines spatial weighting that adapts the model to accurately forecast both demand and supply-demand gap and also indicates the contribution of the corresponding input features for spatio-temporal forecasting.

5. A deep multi-task learning architecture is developed in this study for simultaneously forecasting multiple spatio-temporal forecasting tasks in a city as well as across cities by developing gated ensemble of mixture of experts containing convolutional recurrent neural network (CRNN), convolutional neural network (CNN), and recurrent neural network (RNN).

6. An input agnostic feature weighting layer is developed and integrated with the multi-task learning architecture, which determines weighting that assists in learning a joint representation for different tasks and aids in interpreting the multi-task learning models by indicating the contribution of the input features for prediction.

1.5 Outline of Thesis

The thesis is organized into six chapters. After the introduction in the first chapter, the other five chapters will cover the following topics:

Chapter 2 - Literature Review

In this chapter, previous studies on forecasting demand in the taxi and ride-hailing system are reviewed, which is followed by a review of the previous studies on forecasting supply-demand gap in ride-hailing system. Finally, a review of the studies that applied multi-task learning in taxi/ride-hailing system is provided. Important information and finding from these studies are also discussed.

Chapter 3 – Developing Spatio-temporal Deep Learning Baseline for Ride-hailing System

Chapter three provides a discussion of the methodology and development of spatio-temporal deep learning baseline for ride-hailing system, followed by experiments with real-world data, model and feature ablation, and sensitivity analysis.

Chapter 4 – Developing Spatio-temporal Deep Learning Architecture for Ride-hailing Data with Anonymized Spatial Adjacency Information

Chapter four discusses the methodology and development of the spatio-temporal deep learning architecture for forecasting demand and supply-demand gap in a ride-hailing system with anonymized spatial adjacency information, followed by experiments with real world data, model ablation and interpretation, and sensitivity analysis.

Chapter 5 - Developing Spatio-temporal Multi-task Learning Architecture for Ride-hailing System

This chapter provides a discussion of the methodology and the development of spatio-temporal multi-task learning for simultaneous forecasting of multiple spatio-temporal tasks in

ride-hailing System, followed by experiments with real-world data, model ablation and interpretation, and sensitivity analysis.

Chapter 6 - Conclusions and Recommendations

This chapter concludes the study by summarizing the results and findings from the models developed in this study. Furthermore, the limitations of the study and directions for future research are also provided.

CHAPTER 2: LITERATURE REVIEW

The Literature review of the thesis is organized into three sections. First, a discussion of the previous studies on forecasting demand in the taxi and ride-hailing system are provided. Second, a review of the previous studies on forecasting supply-demand gap in ride-hailing system is provided. Finally, a review of the studies that applied multi-task learning in taxi/ride-hailing system is discussed.

2.1 Demand Forecasting in Taxi/Ride-hailing System

There has been a long tradition of online taxi-hailing demand forecasting research, and several models are developed to date. Due to the recent release of some ride-hailing and taxi datasets, a large amount of studies on spatio-temporal taxi/ride-hailing demand forecasting has also been conducted in the last few years.

2.1.1 Time Series Forecasting Methods

Time series forecasting techniques have been utilized in some of the earlier studies for taxi demand forecasting. There are several traditional methods in time-series data mining, such as auto-regressive (AR), moving average (MA), and auto-regressive moving average (ARMA), etc. Moreover, some modified models such as autoregressive integrated moving average (ARIMA) is introduced. X. Li et al. [58] designed a modified ARIMA in an attempt to capture temporal periodicities with the short-term regular temporal patterns from the univariate time series data of historical taxi demand. The mathematical decomposition of their model further demonstrated that randomness in demand is due to contextual factors (e.g., weather condition, weekday/weekend). Moreira-Matias et al. [5] developed an ensemble of ARIMA and time-varying Poisson models to explicitly deal with short-, medium-, and long-term temporal patterns in the historical taxi demand data. However,

exogenous dependencies were not considered in these studies. To that end, linear regression combined with regularization was applied to features extracted through extensive feature engineering from taxi trip records and external datasets (e.g., point of interest (POI), weather condition, events information, traffic congestion) [59], [60]. Furthermore, Chang et al. [61] performed historical data mining of weather, time and taxi car location using clustering algorithms to forecast taxi demand distribution. These studies showed that forecasting error reduces with the inclusion of relevant additional factors. However, all of these abovementioned studies applied the same model to all spatial units without considering the spatial dependencies, which leads to unreliable demand forecast in some spatial units due to predictability heterogeneity [62].

2.1.2 Machine Learning Methods

In recent years, machine learning has been used widely by the researchers for ride-hailing demand forecasting problems because of its ability to learn complex features by processing massive datasets. Liu et al. [63] extracted important features with the least absolute shrinkage and selection operator (LASSO) that were utilized in random forest and support vector machines (SVM) for forecasting the short-term ride-hailing demand. However, these machine learning techniques do not explicitly model the spatio-temporal dependencies. To improve this limitation, hybrid modeling approaches were found useful. Wei et al. [64] processed spatially related features through an artificial neural network (ANN) and combined them with temporal features based on demand fluctuations for forecasting the ride-hailing demand through gradient boosting decision tree (GBDT). Their study successfully incorporated the spatial correlations in demand based on the similarity of neighbouring zones, however, ignoring temporal dependencies in demand. Y. Li et al. [65] decomposed the ride-hailing demand time series into various frequency series through wavelet analysis, a signal processing technique, and trained SVM on the decomposed signals for forecasting the future ride-hailing demand by recomposing the predicted signals. Although their model successfully

captured the non-stationarity in the time series of a zone, they disregarded the spatial dependencies among the zones. To consider spatial and temporal dependencies simultaneously, Qian et al. [66] combined boosting methods with the Gaussian conditional random field model, which can be used to predict ride-hailing demand by predicting demand distributions. However, spatio-temporal dependencies were modelled only based on historical demand, which limits the detection of various patterns in spatio-temporal dependencies.

2.1.3 Deep Learning Methods

Deep learning, a branch of machine learning, offers more flexibility to design sophisticated architectures for capturing spatial and temporal dependencies in the taxi/ride-hailing demand prediction. For example, Liu et al. [67] used an attention-based deep ensemble net to enhance spatio-temporal prediction accuracy of city-wide ride-hailing demand. Some of the studies [68], [69] applied variants of RNN, especially long short-term memory (LSTM) network [45], for capturing the temporal dependencies in taxi/ride-hailing demand forecasting. Recent studies [70], [71] processed the historical taxi/ride-hailing demand of different zones in a city as a sequence of images and the value of historical demand for a zone in a specific time interval was considered as a pixel value. By learning spatial features and correlations among the images through CNN, the model predicts the future taxi/ride-hailing demand. However, none of these studies was able to capture both spatial and temporal dependencies simultaneously, since CNN is unable to consider temporal patterns and RNN is unable to consider spatial patterns. To consider both spatial and temporal dependencies in a unified framework, Shi et al. [47] developed convolutional LSTM for precipitation forecasting by combining CNN and LSTM in an architecture that modified the LSTM cell by applying convolutions in the tensor-based LSTM cell. Extending from their work, some recent studies developed an ensemble of convolutional LSTMs [4] and attention-based convolutional LSTM [72] for taxi/ride-hailing demand prediction task. Yao et al. [46] predicted ride-hailing

demand forecasting by combining ride-hailing demand similarity-based graphs with local CNN and LSTM.

Later, Graph Convolutional Network (GCN) is found to be an efficient tool to capture region-based relationships through convolution on graphs [9]. Apart from this, Bai et al. [73] developed a graph convolutional recurrent neural network (ConvRNN) to capture both spatio-temporal correlations of passenger demand within a city. Additionally, it incorporated an encoder-decoder module to fuse graph ConvRNN and LSTM to predict the future passenger demand. Similar kind of research was also conducted by Y. Zhou et al. [74], however, their model did not consider graph model, instead, they used deep neural network with attention mechanism and kernel density estimation to predict ride-sourcing demand in the specified city region and time interval. Moreover, Y. Wang et al. [75] used OD matrix to predict ride-sourcing passenger demand using graph convolutions. Also, Y. Xu and Li [76] proposed graph and time-series learning model to capture ride-sourcing demand where the former learns spatial dependencies and the later captures temporal correlations. To tackle the problems lying in short-term spatial and temporal dynamics and long-term temporal dynamics in ride-hailing demand, Jin et al. [77] proposed a graph model, called Deep Multi-view spatio-temporal Virtual Graph Neural Network. This model experimented in New York showed high predictability power in city-wide ride-hailing demand prediction problem. In another study of Pian and Wu [78], they proposed a deep learning model using graph attention network that could predict ride-sourcing demand by handling spatial information of multiple connected regions. Most of the ride-hailing demand forecasting models generally utilized region-based situation awareness or station-based graph representation to capture spatial ride-hailing dynamics in a city. However, Jin et al. [79] utilized both simultaneously by combining GCN, variational encoders and sequence to sequence learning to learn and predict spatio-temporal ride-hailing dynamics.

2.2 Supply-demand Gap Forecasting in Ride-hailing System

While numerous studies focused on only taxi/ride-hailing demand forecasting, very few studies have attempted to explore ride-hailing supply-demand gap mechanism and forecasting. Some of these studies applied machine learning, while others utilized deep learning techniques.

2.2.1 Machine Learning Methods

Some of the previous studies [80], [81] utilized an ensemble of various machine learning algorithms for forecasting the region-wise supply-demand gaps. R. Wang [81] applied a multi-layer ensemble of various machine learning algorithms (e.g., support vector machine, single XGBoost, bagging XGBoost, random forest, etc.). The ensembles of various types of models were used to ensure diversity and stability in prediction, which helped to reduce noise and overfitting. X. Zhang et al. [80] applied a double ensemble technique to combine different GBDT models for forecasting supply-demand gaps in data-sparse situations. Ling et al. [82] proposed an ensemble method by combining SVM and ANN using GPS data to predict supply-demand forecasting. Along with predictive power, this two-stage model could be able to capture the mobility pattern too. However, none of these studies explicitly modelled spatial and temporal dependencies for supply-demand gap forecasting task.

2.2.2 Deep Learning Methods

Deep learning techniques were applied in a few recent studies for supply-demand gap forecasting. For instance, Based on a deep residual network [83], D. Wang et al. [7] developed a deep learning architecture for forecasting the supply-demand gap of ride-hailing services. Their framework is extendible to include new features and requires less amount of feature engineering. Besides, they used an embedding technique for capturing the similarity

in the supply-demand gap patterns. Moreover, J. Li and Wang [84] proposed LSTM to forecast supply-demand gap with the incorporation of weather and traffic information as well as point of interest. Ke et al. [8] took advantage of improved partitioning by developing different types of hexagon-based CNN for different types of mapping functions (square, parity, and cube), and utilized a hexagon-based ensemble technique for forecasting supply-demand gaps. Also, Said and Erradi [85] proposed an approach for the purpose of supply-demand gap forecasting by not to depend on the raw data but to develop new predictors from the raw gap data using multi-view topological data analysis. This approach was found effective and strong-in-power in predicting both supply and demand and their underlying gap with both clean and noisy data. Moreover, recently, Z. Zhang et al. [86] used fuzzy features in combination with deep learning and attention mechanism to capture ride-hailing supply-demand differences both spatially and temporally.

2.3 Multi-task Learning for Spatio-temporal Forecasting in Taxi/Ride-hailing System

For accurate prediction of demand, various researchers attempted to develop multi-task learning architecture using deep learning. However, the reasons behind utilizing multi-task learning were to tackle correlation among different regions both spatially and temporally and to capture the influence of confounding factors (e.g., weather).

Bai et al. [87] proposed an end to end multi-task deep learning with historical data by utilizing CNN to capture spatial correlations with high prediction accuracy of passenger demand. K. Zhang et al. [88] proposed another multi-task learning method with dynamic time warping algorithm that could capture temporal passenger demand in a multi-zone level in China. Moreover, Feng et al. [89] considered the Markov decision process framed by reinforcement learning (RL) to tackle passenger demand reallocation problem. On the other hand, Kim et al. [90] developed a step-wise model by combining econometrics models with

deep learning where the former could be able to interpret demand and the later could increase the predictive power.

Kuang et al. [91] developed an attention-based LSTM to fuse spatio-temporal historical data with 3D ResNet to predict taxi demand for pick-up and dropping-off, simultaneously. K. Zhang et al. [88] also proposed a similar kind of study, however, they proposed multi-task learning with 3-parallel LSTM layers that could co-predict pick-up and dropping-off ride-hailing demand at a time. Luo et al. [92] also proposed multi-task deep learning framework that outperformed the predictive power of single task deep learning, LSTM, SVM and k-nearest neighbors. This study was unique from other studies as the proposed model could capture causality of demand in various traffic zones along with its multi-zonal predictability power.

This study is unique from all the aforementioned studies in the following aspects. Firstly, all of the studies that utilized two-dimensional convolution in their architecture assumed that spatial dependencies among the zones are similar to that among image pixels. However, none of them investigated the effectiveness of preserving the spatial structure for detecting spatial patterns, which is going to be explored in this study. Secondly, the studies that utilized spatio-temporal deep learning techniques were focused on developing complex models without justifying with respect to a spatio-temporal deep learning baseline, while one of the focus of this study is to develop a computationally efficient spatio-temporal deep learning baseline. Thirdly, previous studies designed their models to deal with spatial and temporal dependencies for forecasting taxi/ride-hailing demand and ride-hailing supply-demand gap in a usual scenario with known spatial adjacency information, while this study deals with spatio-temporal dependencies in ride-hailing data with anonymized spatial adjacency information. Fourthly, this study addresses the problem of interpretability in the spatio-temporal deep learning models for spatio-temporal forecasting by utilizing spatial weighting through the feature importance layer, which was not done previously. Fifthly, all of the previous studies

utilized multi-task learning in their architecture to deal with tasks in a city, while the proposed deep multi-task learning architecture in this study deals with simultaneous forecasting of multiple spatio-temporal forecasting tasks in a city as well as across cities, leveraging useful information from the considered spatio-temporal forecasting tasks to develop a shared representation for better generalizations across all considered spatio-temporal forecasting tasks. Finally, this study addresses the problem of shared representation in deep multi-task learning by utilizing a feature weighting layer that emphasizes input features according to their strength and can deal with varying dimensions of input features.

CHAPTER 3: DEVELOPING SPATIO-TEMPORAL DEEP LEARNING BASELINE FOR RIDE-HAILING SYSTEM

This chapter provides a discussion on the development of computationally efficient online taxi-hailing demand forecasting baseline. The methodology and mathematical explanation of the proposed spatio-temporal deep learning baseline architecture are discussed, followed by an evaluation of the proposed architecture with real-world datasets of Didi and comparison of the results with the benchmark models. Furthermore, model and feature ablation of the proposed architecture is conducted. Finally, a sensitivity analysis of the spatio-temporal deep learning models considered in this chapter is provided.

3.1 Preliminaries and Problem Statement

In this section, descriptions of the variables utilized as well as the formulation of the online taxi-hailing demand forecasting problem are provided. Spatio-temporal forecasting tasks in online taxi-hailing system involve time-series data, therefore, historical values of the variables that evolve with time are important indicators for predicting targets. Additionally, POI, time of day, and day of week also have effects on the spatio-temporal forecasting.

Space-time partitioning: For aggregating the variables, the study area is divided into N non-overlapping uniformly sized zones $P = \{1, 2, 3, ..., N\}$ and total time is divided into T time-slots $I = \{1, 2, 3, ..., T\}$ of m minutes interval. The rest of the variables are explained based on this space-time partitioning.

Spatio-temporal variables: Spatio-temporal variables vary simultaneously in the spatial and temporal dimension. The following types of the spatio-temporal variables are utilized in this study:

(1) Demand: The online taxi-hailing demand at all zones during the time-slot $t \in I$ is represented by the vector $\boldsymbol{D}_t \in \mathbb{R}^N$, where $D_{t,p}$ is the total number of successful ride-hailing requests emerging from zone $p \in P$, and $D_{t,p} \in [0, +\infty)$.

(2) Speed: The average speed of the floating taxis of a spatial zone in a time-slot is termed as the speed in that spatial zone. The speed at all zones during the time-slot t is denoted by the vector $\boldsymbol{S}_t \in \mathbb{R}^N$. The speed at a zone p during the time-slot t is denoted by $S_{t,p}$.

(3) Accessibility: The accessibility of a spatial zone in a time-slot is measured by the number of taxis with passengers crossing that spatial zone. The accessibility of all zones in the time-slot t is expressed by the vector $\boldsymbol{M}_t \in \mathbb{R}^N$. The accessibility of a zone p during the time-slot t is denoted by $M_{t,p}$.

Weather variables: The weather variables include the meteorological features that vary randomly across time, but not space. For maintaining consistency of input dimensions in the proposed architectures, these variables may have to be repeated across the zones by utilizing the repeating function $f_{RZ}(\cdot; N): \mathbb{R}^{1\times M} \to \mathbb{R}^{N\times M}$, where M represents the number of feature categories. The following types of weather variables are included in this study:

(1) Categorical weather variables: For the time-slot t, the weather conditions are the categorical weather variables, denoted by the row vector $\boldsymbol{wc}_t \in \mathbb{R}^{1\times C}$ consisting of C weather categories (e.g., sunny, rainy, snowy, etc.), where each weather category is encoded by one-hot encoding, i.e., $wc_{c,t} \in \{0,1\}$.

(2) Continuous weather variables: For the time-slot t, a continuous weather variable is expressed by the vector $\boldsymbol{wt}_t \in \mathbb{R}$. The continuous weather variables utilized in this study are temperature, particulate matter, dew point, humidity, cloud cover, wind speed, and visibility.

Context variables: The context variables are either periodic or fixed across time. For spatio-temporal forecasting, these variables are either repeated across the zones by applying the repeating function f_{RZ} or repeated across the time-slots by utilizing the repeating function $f_{RT}(\cdot;T):\mathbb{R}^N \to \mathbb{R}^{N\times 1\times T}$. These are as follows:

(1) Temporal context: Following [4], exploratory data analysis of the trends in demand and supply-demand gap revealed two types of periodic contexts: time-of-day and day-of-peak. A time-slot t of a day belongs to one of the three 8-hour time-of-day intervals: sleep (first 8-hour), peak (mid-8-hour), and off-peak (last 8-hour), denoted by the row vector $\boldsymbol{cd}_t \in \mathbb{R}^{1\times 3}$, where each interval i is one-hot encoded, i.e., $cd_{i,t} \in \{0,1\}$. Furthermore, a time-slot t falls into one of the day-of-week categories: weekday or weekend, represented by the vector $\boldsymbol{cw}_t \in \mathbb{R}$, where $cw_t \in \{0,1\}$. The temporal context vectors $\boldsymbol{cd}_t$ and $\boldsymbol{cw}_t$ are repeated across the zones to form the time-of-day matrix $\boldsymbol{CD}_t \in \mathbb{R}^{N\times 3}$ and the day-of-week vector $\boldsymbol{CD}_t \in \mathbb{R}^N$ respectively.

(2) Spatial context: The spatial context refers to the number of POIs across the zones, denoted by the vector $\boldsymbol{cp}_p \in \mathbb{R}^N$, which is fixed across time. Therefore, it is repeated across the time-slots with the repeating function f_{RT} to form the spatial context vector for the time-slot t, represented by the vector $\boldsymbol{CP}_t \in \mathbb{R}^N$.

With the abovementioned definition of the variables, the spatio-temporal forecasting problem can be formulated as follows:

Problem Statement: Given, the historical data of spatio-temporal variables and weather variables up to bth previous time-slot starting from $t-1$, and known data of context variables at the time-slot t, it is required to predict the ground truth vector $\boldsymbol{A}_t$ at the time-slot t for spatio-temporal demand forecasting in an online taxi-hailing system.

3.2 Methodology

This section provides a brief description of the main components (i.e., CNN, RNN, and ConvRNN) utilized in developing the spatio-temporal online taxi-hailing demand forecasting baseline architecture in this study.

3.2.1 Convolutional Neural Network

The CNN is a specialized neural network for detecting spatial dependencies by various types of filters (i.e., weights and bias) sliding over and convolving with the input. To detect same kind of spatial features with a filter, parameters of a filter are shared across the zones/pixels. The output feature vector of a filter k in a one-dimensional CNN layer can be expressed as follows:

$$\boldsymbol{Z}^{(k)} = \sigma\left(\boldsymbol{X}_t * \boldsymbol{W}^{(k)} + \boldsymbol{b}^{(k)}\right) \tag{3.1}$$

where $*$ is one-dimensional convolution operator, $\boldsymbol{X}_t$ is the input to be convolved with the filter k, $\boldsymbol{Z}^{(k)}$ refers to the output feature vector for the filter, $\boldsymbol{W}^{(k)}$ serves as the shared weight matrix of the filter, and $\boldsymbol{b}^{(k)}$ is the shared bias vector of the filter.

However, spatial dependencies in spatio-temporal forecasting not only depends on neighboring zones, but also on distant zones (e.g., functional similarity, transportation connectivity) [9]. Therefore, a one-dimensional CNN is utilized in this study for developing the proposed baseline architecture, based on recent findings that showed the applicability of CNN to learn from scrambled images [50]. The one-dimensional CNN filters are slid over only across the flattened spatial dimension of the input with spatio-temporal feature columns. As a result, the computational complexity of a one-dimensional CNN layer is less than that of a two-dimensional CNN layer. For a one-dimensional convolution, a single convolution

consists of a dot product of a filter weight $K \in \mathbb{R}^{c \times d}$ with a receptive field $R \in \mathbb{R}^{c \times d}$, where c is the length of the filter and d is the depth of the filter, i.e., dimensionality of the feature space. This results in time complexity of $O(d)$ for a single dot product consisting of d multiplications and $d-1$ additions, which goes up to $O(c \cdot d)$ for c rows. At the layer level, since a filter slides over the input with length n for $n-c+1$ times, the complexity of f filters can reach up to $O(n \cdot c \cdot d \cdot f)$ considering $n \gg c$. However, for a two-dimensional convolution, a single convolution consists of a dot product of a filter weight $K \in \mathbb{R}^{c \times c \times d}$ with a receptive field $R \in \mathbb{R}^{c \times c \times d}$, which will eventually result in $O(n \cdot c^2 \cdot d \cdot f)$ at the layer level. Therefore, the time complexity of a one-dimensional CNN will be reduced by c times.

3.2.2 Long Short-term Memory Network

The RNN is an exclusive architecture for detecting temporal dependencies from time-series data where the features of one time-slot are correlated with the features of the previous time-slots. In comparison to the non-recurrent connections in conventional neural networks, the RNN has recurrent connections, i.e., the outputs of the hidden layer neurons from the previous time step of a sequence are utilized with the inputs of the current time step, which can be expressed by Eq. (3.2):

$$\boldsymbol{h}_t = \sigma(\boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{W}\boldsymbol{h}_{t-1} + \boldsymbol{b}) \tag{3.2}$$

where $\boldsymbol{x}_t \in \mathbb{R}^F$ is the input vector at the current time step containing F features, $\boldsymbol{h}_t \in \mathbb{R}^H$ and $\boldsymbol{h}_{t-1} \in \mathbb{R}^H$ refer to the output vectors of size H representing hidden layer neurons of current and previous time step respectively, $\boldsymbol{U} \in \mathbb{R}^{H \times F}$ and serve as the weights for the input of the current step and the outputs of the previous step respectively, and $\boldsymbol{b} \in \mathbb{R}^H$ is the bias vector.

However, repeated multiplication of the recurrent hidden layer weight matrix during training results in the vanishing/exploding gradient problem in RNN that limits the storing of long-term information, which can be tackled through LSTM [45] by including additive updates in the hidden layer. The LSTM cell, as shown in Figure 3.1, modifies the recurrent structure in Eq. (3.2) through Eq. (3.3)-(3.8):

$$\boldsymbol{i}_t = \sigma\left(\boldsymbol{U}^{(i)}\boldsymbol{x}_t + \boldsymbol{W}^{(i)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(i)}\right) \tag{3.3}$$

$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{U}^{(f)}\boldsymbol{x}_t + \boldsymbol{W}^{(f)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(f)}\right) \tag{3.4}$$

$$\boldsymbol{o}_t = \sigma\left(\boldsymbol{U}^{(o)}\boldsymbol{x}_t + \boldsymbol{W}^{(o)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(o)}\right) \tag{3.5}$$

$$\tilde{\boldsymbol{c}}_t = \tanh\left(\boldsymbol{U}^{(c)}\boldsymbol{x}_t + \boldsymbol{W}^{(c)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(c)}\right) \tag{3.6}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t \tag{3.7}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh\left(\boldsymbol{c}_t\right) \tag{3.8}$$

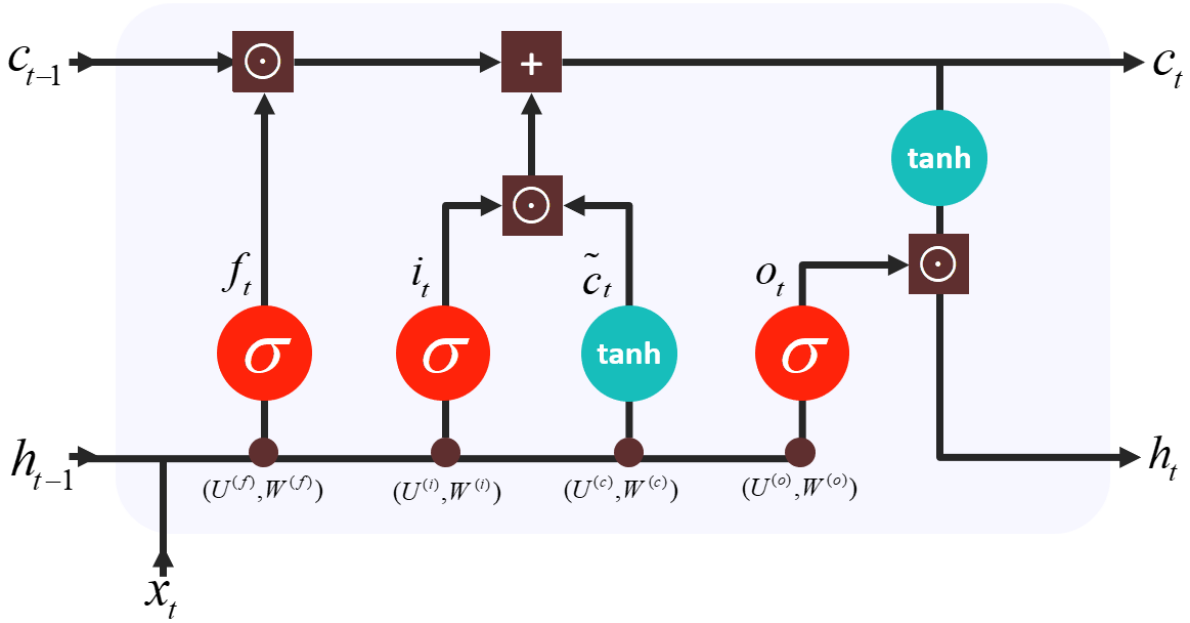


Figure 3.1. An LSTM Cell

(Source: Modified from: [93])

where $\boldsymbol{i}_t$, $\boldsymbol{f}_t$, and $\boldsymbol{o}_t$ are input gate, forget gate, and output gate respectively, containing corresponding input weights, recurrent weights, and bias vector; $\tilde{\boldsymbol{c}}_t$, $\boldsymbol{c}_t$, and $\boldsymbol{h}_t$ are the update values, cell state, and output hidden states respectively.

3.2.3 Convolutional Recurrent Neural Network

Convolutional recurrent layers are exclusively utilized to detect spatial and temporal dependencies simultaneously from spatio-temporal features. To prevent vanishing/exploding gradient problems in convolutional recurrent layers, convolutional LSTM [47] and gated convolutional recurrent unit [94] have been developed. However, a convolutional recurrent layer with a vanilla RNN (ConvRNN) can also tackle the vanishing/exploding gradient problem by utilizing a linear activation function, while being computationally cheaper. A ConvRNN can be formulated by modifying the cell structure of the vanilla RNN in Eq. (3.2) to take spatio-temporal features as inputs and replacing the matrix multiplication with a convolution operator. A ConvRNN cell, as shown in Figure 3.2, can be expressed as follows:

$$\boldsymbol{H}_t = ReLU\left(\boldsymbol{U}^{(c)} * \boldsymbol{X}_t + \boldsymbol{W}^{(c)} * \boldsymbol{H}_{t-1} + \boldsymbol{b}^{(c)}\right) \tag{3.9}$$

where $\boldsymbol{X}_t$ is the input at the current time step, $\boldsymbol{H}_t$ and $\boldsymbol{H}_{t-1}$ refer to the output of current and previous time step respectively, $\boldsymbol{U}^{(c)}$ and $\boldsymbol{W}^{(c)}$ serve as the filters for the input of the current step and the outputs of the previous step respectively, and $\boldsymbol{b}^{(c)}$ is the bias. It is noteworthy to mention that the proposed architecture utilizes a one-dimensional convolution in Eq. (3.9).

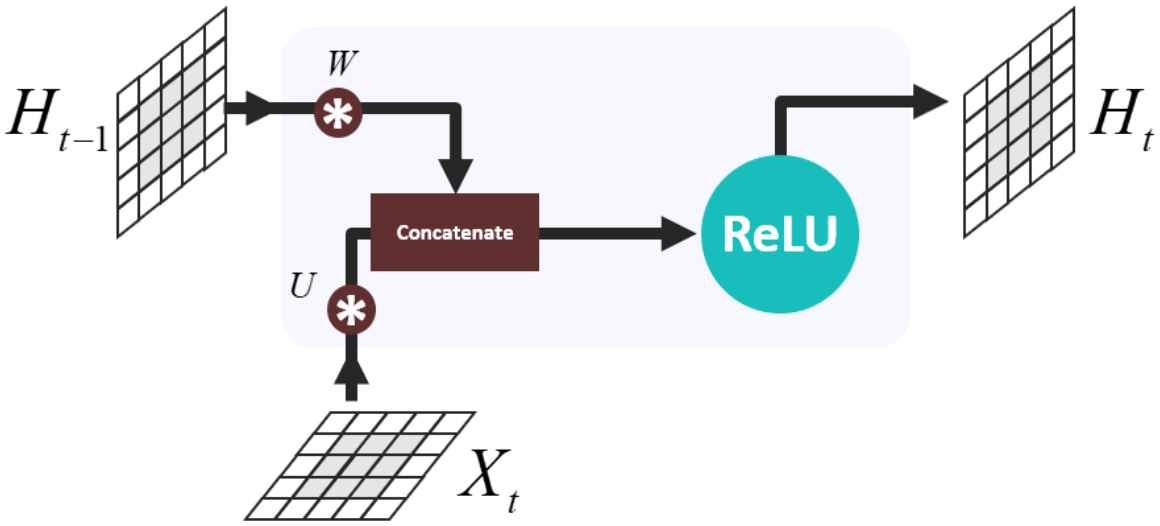


Figure 3.2. A ConvRNN cell

(Source: Modified from [95])

3.2.4 Development of the Spatio-temporal Baseline Architecture

The workflow of the proposed spatio-temporal online taxi-hailing demand forecasting architecture for processing different variables (i.e., spatio-temporal variables, weather variables, and context variables) is illustrated in Figure 3.3. The spatial and temporal features are simultaneously extracted from the spatio-temporal variables by layers of stacked 1D-ConvRNN, which are then concatenated with the spatial context variable before passing through layers of one-dimensional CNN for further extraction of spatial features based on spatial context. At the same time, weather variables are fed into layers of LSTM, followed by the repeating function to copy the extracted feature from LSTM across all the zones since these variables do not vary with space, but only with time. The features extracted from 1D-CNN followed by 1D-ConvRNN and the repeated features extracted from LSTM are concatenated with the temporal context in order to account for the effects of time-of-day and day-of-week, which is then passed through a fully connected layer (i.e., dense layer) to get the prediction output vector $\boldsymbol{O}_t$ that is compared with the ground truth vector $\boldsymbol{A}_t$ for calculating the model loss. Techniques such as repeating function f_{RZ} and concatenation

function $f_{Concatenate}$ (e.g., $f_{Concatenate}(\boldsymbol{P};\boldsymbol{Q}):\mathbb{R}^{N,N}\rightarrow\mathbb{R}^{N\times 2}$, where $\boldsymbol{P}\in\mathbb{R}^{N}$ and $\boldsymbol{Q}\in\mathbb{R}^{N}$ are the vectors to be concatenated) are utilized in the architecture to adapt the input requirement of the architecture.

The training of the proposed architecture involves minimizing the mean squared error between the predicted values and the ground truth, which is achieved through the loss function. The objective of the overall loss function f_{Loss} applied in the architecture including regularization terms can be expressed as Eq. (3.10):

$$\min_{W^{(A)},W^{(FI)},b} f_{Loss}=\left\|\boldsymbol{O}_t-\boldsymbol{A}_t\right\|_2^2+\alpha\left\|\boldsymbol{W}^{(A)}\right\|_2^2 \qquad (3.10)$$

where $\boldsymbol{A}_t\in\mathbb{R}^{N}$ is the ground truth vector for a task p, $\boldsymbol{W}^{(A)}$ refers to all parameters of the proposed architecture, and α is the regularization parameters. The L2-norm of regularization are utilized in the architecture assists in avoiding overfitting issues.

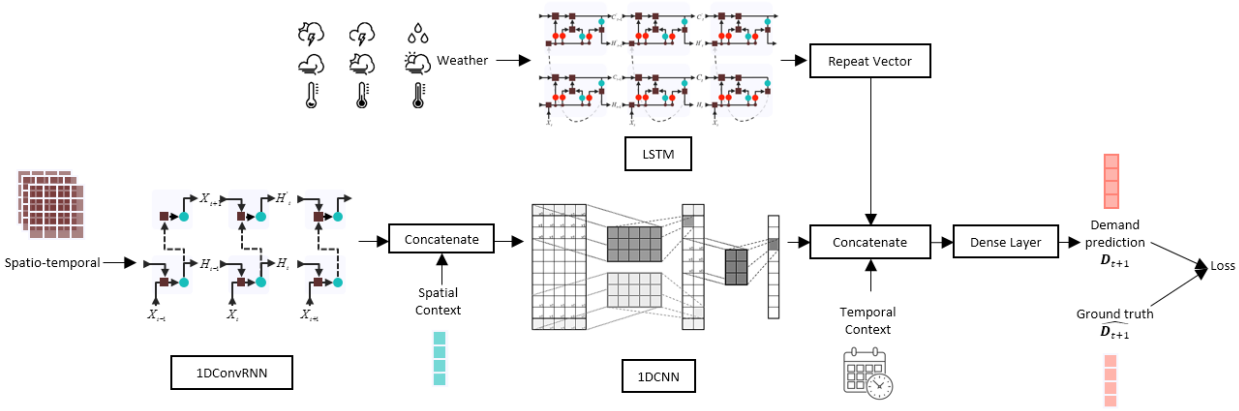


Figure 3.3. Workflow of the spatio-temporal baseline architecture

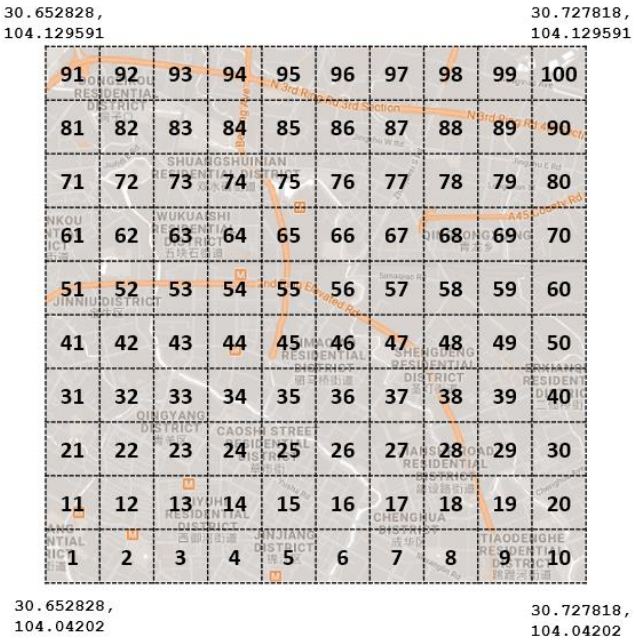
3.3 Data Preparation, Experiments, and Model Evaluation

This section sheds light on the details of the data preprocessing, experiments, and discussion of the results for the proposed spatio-temporal deep learning baseline architecture. Moreover, ablation and sensitivity analysis of the proposed architecture is also provided.

3.3.1 Data Description and Preprocessing

Trajectory datasets from Chengdu under Didi Chuxing GAIA open dataset initiative [96] are selected for experimentation. Since the datasets provided raw trajectories only, data processing is used to extract the spatio-temporal variables and external weather and POI datasets are used to extract the required weather and POI variables.

The city area is divided into 10×10 square zones as shown in Figure 3.. The dataset span from 1st October 2016 to 30th November 2016. Each day in the dataset is divided into 96 time-slots of 15 minutes interval. Furthermore, the total time-slots in a zone are split into training, validation, and testing sets for the experiments. Around 30% of the time-slots are reserved for validation and testing, and the rest of the data are used in training the models.



(a) Chengdu

Figure 3.4. Spatial partitioning of Chengdu

The Chengdu dataset contains anonymized trajectories of around 11.75 million ride-hailing trips. The spatio-temporal variables are extracted from the trajectories by various data processing techniques. The accessibility of a zone in a time-slot is calculated by counting the trajectories that fall in that zone. The average speed of a ride-hailing vehicle in a zone is found by extracting speed of each ride-hailing vehicle, calculating the distance from the trajectory portion that falls in that zone and dividing with the corresponding time to cover that distance. The demand of a zone is calculated by extracting starting point of the trajectories and aggregating them in the respective time-slot. Around 12 percent of the time-slots are found to be zero-demand time-slots. The zone-wise distribution of the total online taxi-hailing demand in Chengdu is shown in Figure 3..

Data scraping is utilized to extract city-level weather variables (i.e., weather category, temperature, humidity, visibility, cloud cover, and wind speed) at 15 minutes interval from Dark Sky [97]. The zone-wise time-invariant POI information for Chengdu are extracted from Gaode Map [98]. The zone-wise number of facilities for different POI types are aggregated to get the total POI per zone.

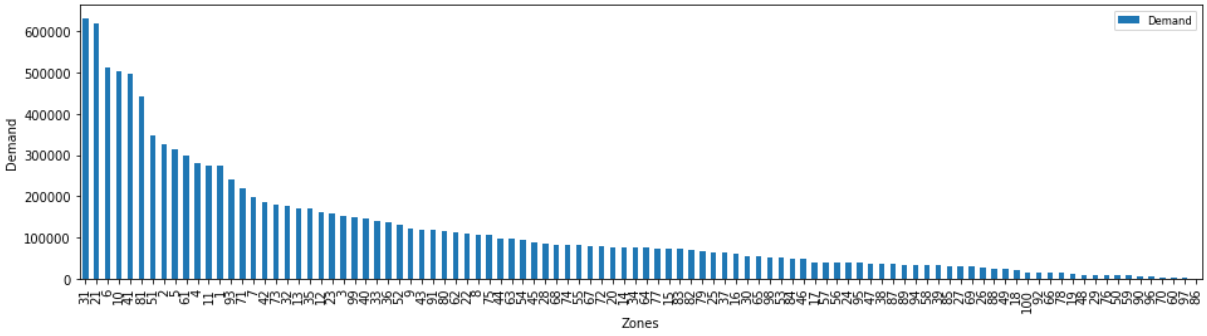


Figure 3.5. Distribution of demand across zones in Chengdu

3.3.2 Model Evaluation

The performance of the proposed architecture is compared to a set of benchmark deep learning and machine learning algorithms. Furthermore, an ablation analysis of the proposed

architecture is conducted to justify the use of each model components. For a fair comparison, all models utilized the same lookback window, i.e., up to sixth previous time-slot. The configuration of the proposed architecture is decided by fine-tuning of the hyperparameters. The finalized settings of the hyperparameters are presented in Table 3.1.

Table 3.1. Hyperparameter settings of the proposed spatio-temporal baseline

	Hyperparameter settings
(a) 1DConvRNN	Layers: 2; filters: 100 in 1st layer; 200 in 2nd layer; filter length: 3 Weight initialization: uniform; activation: ReLU
(b) 1DCNN	Layers: 2; filters: 100 in 1st layer; 200 in 2nd layer; filter length: 3 Weight initialization: uniform; activation: ReLU
(c) LSTM	Layers: 2 layers; hidden units: 4 units per layer Weight initialization: uniform; activation: tanh
(d) Dense	Weight initialization: uniform; activation: ReLU
(e) Training	Optimizer: Adamax [99]; learning rate: 0.001; batch size: 32 Regularization: L1 = 0.001, L2 = 0.001; early stopping: patience = 20 epochs

In order to assess the performance of the proposed architecture as a baseline, following deep learning models with increased complexity are considered in this study, which processes the zone-wise historical spatio-temporal data and spatial context data as images pixels:

1) 2DConvRNN + 2DCNN + LSTM: This architecture is similar to the proposed baseline architecture except that two-dimensional convolution is utilized in ConvRNN and CNN.

2) 2DConvLSTM + 2DCNN + LSTM: This architecture utilizes 2D-ConvLSTM and 2D-CNN in place of 1D-ConvRNN and 1D-CNN of the proposed baseline architecture respectively.

It is noteworthy to mention that, hyperparameters of the aforementioned deep learning benchmarks are also tuned to get the best performance in order to provide a fair comparison.
In addition to the abovementioned deep learning models, several machine learning models are also considered, which are extensively tuned by utilizing automated machine learning frameworks. They are as follows:

1) Gradient Boosting Machine (GBM): The GBM [100] is an ensemble method that is built upon several additive regression trees through the utilization of the gradient descent technique.

2) Extreme Gradient Boosting (XGBoost): The XGBoost [101], a modified version of GBM, is a popular algorithm that has topped in many machine learning competitions. The XGBoost algorithm utilizes regularized gradient boosting to control overfitting.

3) Random Forest (RF): The RF [102] is an ensemble method that utilizes several weak learner regression trees. The splitting of the trees is determined by finding the most appropriate discriminative threshold for a subset of features.

4) Extremely Randomized Trees (XRT): The XRT [103] is similar to the RF except that it extracts randomly generated thresholds for each feature and the best threshold is utilized for splitting the trees.

5) Generalized Linear Model (GLM): The GLM [104] is a generalization of the linear regression that allows any of the exponential family of distributions in the errors of the

outcome variable. Since the output is continuous, the GLM implementation in this study assumes Gaussian distribution.

6) Artificial Neural Network (ANN): The ANN [105] contains a feedforward neural network with several hidden layers to learn hierarchical representation from the input features. The ANN utilized in this study is trained with a mini-batch size of 1.

The performances of the models utilized in this study are evaluated with three metrics: mean absolute error (MAE), root mean squared error (RMSE), and symmetric mean absolute percentage error (sMAPE), which can be computed by using Eq. (3.11)-(3.13):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|\boldsymbol{O}_i - \boldsymbol{A}_i\right| \tag{3.11}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{O}_i - \boldsymbol{A}_i\right)^2} \tag{3.12}$$

$$sMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|\boldsymbol{O}_i - \boldsymbol{A}_i\right|}{\left|\boldsymbol{O}_i\right| + \left|\boldsymbol{A}_i\right| + 1} \tag{3.13}$$

where $\boldsymbol{O}_i$ and $\boldsymbol{A}_i$ are the predicted vector and ground truth vector, respectively, at time-slot i in the test set with size n time-slots. Since the target value contains zero and sMAPE produces inaccurate statistics when encountered with zero, therefore, a modified sMAPE [5] is utilized.

The proposed architecture is trained on a server with 4 Core (hyper-threaded) Xeon processor (2.30 GHz), 25 GB RAM, and a Tesla P-100 GPU. The GESME-Net and its variations are written in Python 3 using Keras [106] with Tensorflow [107] backend. All the machine learning benchmarks are implemented in H2O AutoML [108].

The performances of the proposed baseline architecture and the benchmark models are reported in Table 3.2. From the comparison among the deep learning models, it is evident that

model from the proposed architecture based on one-dimensional convolution marginally outperforms both deep learning benchmarks. Although the sMAPE of all the deep learning models are almost same, model from the proposed architecture provides about 1 % improvement in RMSE than the deep learning benchmarks. It is clear that model performance metrics does not deteriorate if one-dimensional CNN is applied to historical data with flattened zones rather than utilizing a two-dimensional CNN on historical data that preserved spatial structure. Furthermore, although there is a large improvement in training time in the machine learning models, the performance drop in comparison to the deep learning benchmarks are not trivial. The model from the proposed architecture has about 10 % lower RMSE and 3 % lower sMAPE than the best machine learning benchmark GBM.

Table 3.2. Performances evaluation of proposed spatio-temporal baseline and benchmark models

Model	MAE	RMSE	sMAPE	Time (s)
1DConvRNN + 1DCNN + LSTM	3.6385	5.8324	0.1513	648.754
2DConvRNN + 2DCNN + LSTM	3.6572	5.8844	0.1525	1133.821
2DConvLSTM + 2DCNN + LSTM	3.6612	5.8982	0.1510	839.004
GBM	4.0589	6.4037	0.1805	20.799
XGBoost	4.1542	6.5277	0.1831	27.680
XRT	4.3208	6.7635	0.1889	133.889
ANN	4.3187	6.8317	0.2009	22.217
DRF	4.5126	6.9150	0.2000	114.411
GLM	4.3403	7.0395	0.1843	1.679

Ablation analysis of the proposed architecture is also conducted by removing the model components and the feature categories one at a time. The results of ablation analysis are presented in Table 3.3. For ablation of model components, around 2 % increase the RMSE is found due to removal of the components, which indicates that each of the model components is essential for achieving the best performance from the proposed architecture. The ablation of context and weather features shows similar deterioration in model performance, which shows that the extraction of these features from external datasets can improve model performance. Most importantly, substantial deterioration is seen due to removal of spatio-temporal features, about 113 % increase in the RMSE and about 18 % increase in sMAPE, indicating that the spatio-temporal features are the most important ones for spatio-temporal online taxi-hailing demand forecasting.

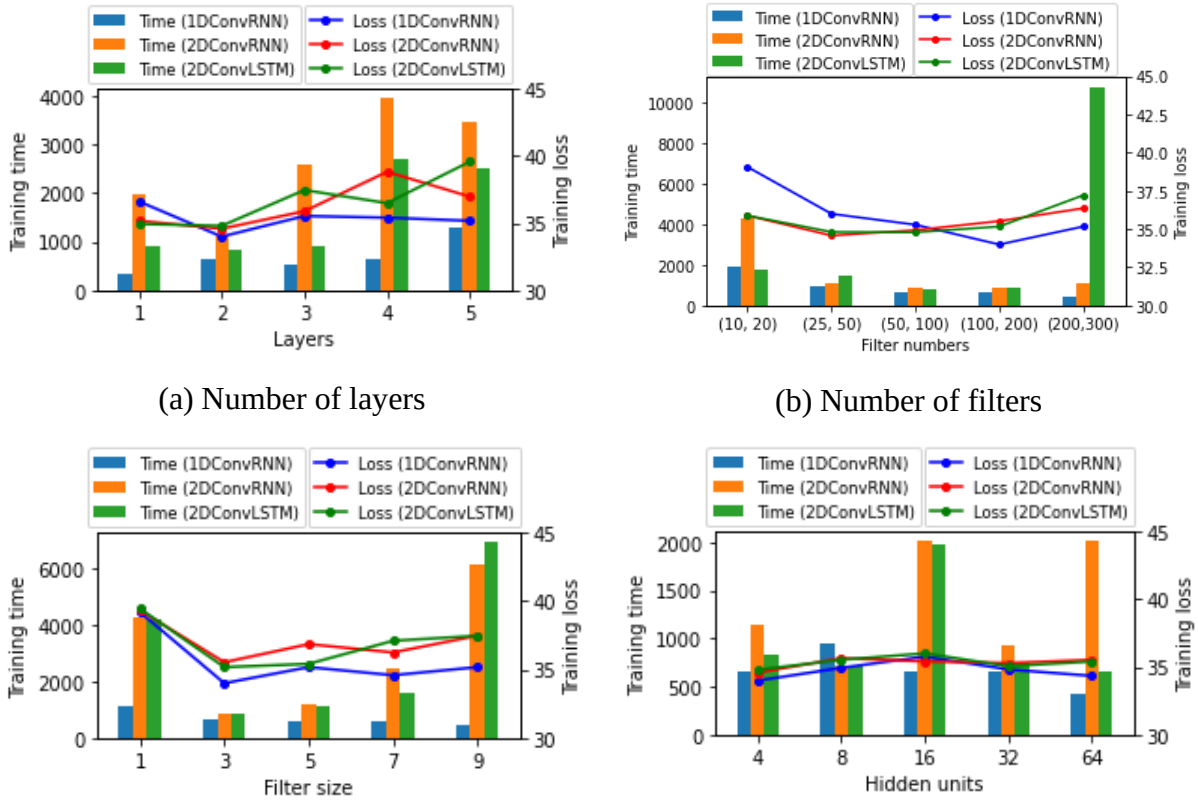
Table 3.3. Ablation analysis of the proposed spatio-temporal baseline

Ablation	Component	MAE	RMSE	sMAPE	Time (s)
Model	1DConvRNN	3.7182	5.9319	0.1557	630.932
	1DCNN	3.7607	5.9779	0.1601	365.121
	RNN	3.7098	5.9225	0.1538	363.096
Feature	Spatio-temporal	7.5074	12.4450	0.3345	1392.914
	Context	3.7301	5.9476	0.1570	436.7743
	Weather	3.7320	5.9398	0.1568	405.9621

3.3.3 Sensitivity Analysis

A sensitivity analysis of the deep learning models considered in this study is conducted in terms of four hyperparameters: number of layers, number of filters in CNN, filter size of CNN, and the number of hidden units in LSTM. In order to get the best performance for

every hyperparameter configuration, all the experiments on hyperparameter tuning are conducted with early stopping.



(a) Number of layers

(b) Number of filters

(c) Filter size

(d) LSTM hidden units

Figure 3.6. Sensitivity analysis of the deep learning architectures

The variation of the training loss (i.e., mean squared error) of the deep learning models with respect to the number of layers is shown in Figure 3. (a). Losses for all the deep learning models reach the lowest for 2 layers and then rise again with an increased number of layers. This result is not unintuitive since it is theoretically proven that neural networks with 2 hidden layers can approximate any continuous function [109]. However, a consistent loss with lowest training time is seen for the proposed architecture in comparison to the deep learning benchmarks for higher number of layers, which indicates the suitability of the proposed architecture as a baseline for developing deeper models.

The performance of the deep learning models with respect to the number of filters in the first and second layer of the convolution utilized in ConvRNN and CNN are shown in Figure 3. (b). It is evident that training losses of the proposed architecture gradually decreases and achieves the minimum for a combination of 100 filters in the first layer and 200 filters in the second layer, utilizing more filters than the deep learning benchmarks but providing the minimum loss and lowest training time in comparison to them. This also indicates that one-dimensional convolutions in the CNN and ConvRNN are detecting more patterns in comparison to the two-dimensional convolution.

Similar effects of filter size on the training losses of the deep learning models are seen in Figure 3. (c). It is noteworthy to mention that filter size refers to one-dimensional filter length in the case of one-dimensional convolution for the proposed architecture, whereas it refers to the length of the two-dimensional square filter in the case of the deep learning benchmarks. The model losses reach the minimum for the filter size 3 in all the models, where lowest model loss and training time are found in the model from the proposed architecture.

Comparing with other hyperparameters, variations of the LSTM hidden units show very small changes in the training losses for all the deep learning models, showing a common best performance achieved by 4 hidden units, as seen in Figure 3. (d). The small variations for the hidden units are most likely due to the fact that most of the pattern detection in these deep learning models are done by the CNN and the ConvRNN/ConvLSTM.

CHAPTER 4: DEVELOPING SPATIO-TEMPORAL DEEP LEARNING ARCHITECTURE FOR RIDE-HAILING DATA WITH ANONYMIZED SPATIAL ADJACENCY INFORMATION

This chapter provides a solution to the ride-hailing demand and supply-demand gap forecasting problem with anonymized spatial adjacency information. A detailed description of the methodology and mathematical explanation of the proposed FOCIR-Net is given in this chapter, followed by evaluation of the proposed architecture with real-world datasets of Didi and comparison of the result with the benchmark models. Furthermore, model ablation of the proposed architecture is conducted. Finally, a sensitivity analysis of the spatio-temporal deep learning models considered in this chapter is provided.

4.1 Preliminaries and Problem Statement

Both ride-hailing demand and supply-demand gap are time-series forecasting tasks, where historical values of related variables can be important indicators for predicting future values. The historical values of the ride-hailing orders, the supplied quantity of ride-hailing cars, and the number of congested roads contain useful information regarding the spatio-temporal dependencies of demand and supply-demand gap. Additionally, weather condition, POI, time of day, and day of week also affects the demand and supply-demand gap of a region.

This section presents explanations of the variables and notations used in this study along with the forecasting problem of demand and supply-demand gap.

***Space-time partitioning*:** For variable aggregation, the study area is divided into N non-overlapping uniformly sized zones $P = \{1, 2, 3, ..., N\}$ and total days are divided into T time-slots $I = \{1, 2, 3, ..., T\}$ of m minutes interval. The rest of the variables are defined based on the space-time partitioning.

***Spatio-temporal variables*:** The features that simultaneously vary in the spatial and temporal dimension are regarded as spatio-temporal variables. The types of the spatio-temporal variables utilized in this study are as follows:

1) Quantity supplied: The quantity of ride-hailing cars supplied at all zones during the time-slot $t \in I$ is represented by the vector $\boldsymbol{S}_t \in \mathbb{R}^N$, where $S_{t,p}$ is the total number of ride-hailing requests at zone $p \in P$ successfully matched with drivers by the ride-hailing platform, and $S_{t,p} \in [0, +\infty)$.

2) Demand: The total number of ride-hailing requests from a zone in a time interval is referred to as the demand, which includes both successfully matched and unanswered ride-hailing requests. The demand of all zones during the time-slot t is placed in the vector $\boldsymbol{D}_t \in \mathbb{R}^N$. The demand at zone p during the time-slot t is denoted as $D_{t,p}$, where $D_{t,p} \geq S_{t,p}$.

3) Supply-demand gap: The total number of unmatched ride-hailing requests from a spatial zone in a time-slot is termed as the supply-demand gap. The supply-demand gap of all zones during the time-slot t is expressed as the vector $\boldsymbol{G}_t \in \mathbb{R}^N$. The supply-demand gap of a zone p during the time-slot t is denoted by $G_{t,p}$, where $0 \leq G_{t,p} \leq D_{t,p}$.

4) Traffic congestion: The traffic congestion of all zones is denoted by the vector $\boldsymbol{TC}_t \in \mathbb{R}^N$, where $TC_{t,p}$ represents the total number of congested roads belonging to a zone p during the time-slot t.

***Temporal variables*:** The temporal variables include the features that vary randomly across time, but not space. For maintaining consistency of input dimensions in the proposed architecture, these variables need to be repeated across the zones by utilizing the repeating

function $f_{RZ}(\cdot;N):\mathbb{R}^{1\times M}\to\mathbb{R}^{N\times M}$, where M represents the number of feature categories. The following categories of temporal variables are included in this study:

1) Weather condition: For the time-slot t, the weather conditions are represented by the row vector $\boldsymbol{wc}_t\in\mathbb{R}^{1\times C}$ consisting of C weather categories (e.g., sunny, rainy, snowy, etc.), where each category is encoded by one-hot encoding, i.e., $wc_{c,t}\in\{0,1\}$. The weather condition vector $\boldsymbol{wc}_t$ is repeated across every zone to produce the weather condition matrix for the time-slot t, denoted by $\boldsymbol{WC}_t\in\mathbb{R}^{N\times C}$.

2) Temperature: The temperature (in ℃) for the time-slot t is repeated across the zones to produce the temperature vector $\boldsymbol{WT}_t\in\mathbb{R}^{N}$.

3) Particulate matter: For the time-slot t, the atmospheric particulate matter, $PM_{2.5}$ is repeated across the zones to form the particulate matter vector $\boldsymbol{WP}_t\in\mathbb{R}^{N}$.

***Context variables*:** The context variables are either periodic or fixed across time. For spatio-temporal forecasting, these variables are either repeated across the zones by applying the repeating function f_{RZ} or repeated across the time-slots by utilizing the repeating function $f_{RT}(\cdot;T):\mathbb{R}^{N}\to\mathbb{R}^{N\times 1\times T}$. These are as follows:

1) Temporal context: Following [4], exploratory data analysis of the trends in demand and supply-demand gap revealed two types of periodic contexts: time-of-day and day-of-peak. A time-slot t of a day belongs to one of the three 8-hour time-of-day intervals: sleep (first 8-hour), peak (mid-8-hour), and off-peak (last 8-hour), denoted by the row vector $\boldsymbol{cd}_t\in\mathbb{R}^{1\times 3}$, where each interval i is one-hot encoded, i.e., $cd_{i,t}\in\{0,1\}$. Furthermore, a time-slot t falls into one of the day-of-week categories: weekday or weekend, represented by the vector $\boldsymbol{cw}_t\in\mathbb{R}$, where $cw_t\in\{0,1\}$. The temporal context vectors $\boldsymbol{cd}_t$ and $\boldsymbol{cw}_t$ are repeated across

the zones to form the time-of-day matrix $\boldsymbol{CD}_t \in \mathbb{R}^{N\times 3}$ and the day-of-week vector $\boldsymbol{CD}_t \in \mathbb{R}^{N}$ respectively.

2) Spatial context: The spatial context refers to the number of POIs across the zones, denoted by the vector $\boldsymbol{cp}_p \in \mathbb{R}^{N}$, which is fixed across time. Therefore, it is repeated across the time-slots with the repeating function f_{RT} to form the spatial context vector for the time-slot t, represented by the vector $\boldsymbol{CP}_t \in \mathbb{R}^{N}$.

With the abovementioned definition of the variables, the spatio-temporal forecasting problem can be formulated as follows:

Problem Statement: Given, the historical data of spatio-temporal variables and temporal variables up to b th previous time-slot starting from $t-1$, and known data of context variables at the time-slot t, it is required to predict the ground truth vector $\boldsymbol{A}_t$ at the time-slot t for spatio-temporal forecasting of demand and supply-demand in ride-hailing system with anonymized spatial adjacency information.

4.2 Methodology

This section provides a brief description of the main components (i.e., feature importance layer, one-dimensional CNN, and zone-distributed IndRNN) as well as the proposed FOCIR-Net architecture.

4.2.1 Feature Importance Layer

Deep learning methods involve capturing complex interaction among features through multiple intermediate layers between inputs and outputs, which makes it difficult to interpret the contribution of the input features towards prediction. To address this issue, a number of

methods [110]–[112] based on a one-to-one linear layer, usually utilized after the input layer, were implemented in deep artificial neural networks. The main idea of a one-to-one linear layer is similar to a fully connected layer except that a neuron in a one-to-one linear layer is connected with only one input rather than all inputs. This study utilizes similar idea of feature importance layer except that the internal structure is modified for spatio-temporal forecasting, as expressed by the following function in Eq. (4.1):

$$f_{FeatureImportance}\left(\boldsymbol{X}_t\right) = \boldsymbol{X}_t \circ \sigma(\boldsymbol{W}^{(FI)}) \tag{4.1}$$

The operator $\circ$ in Eq. (4.1) indicates Hadamard product, i.e., elementwise multiplication, between the input matrix $\boldsymbol{X}_t \in \mathbb{R}^{N\times F}$ for the time-slot t, containing N zones and F input features, and the outputs of the activation function σ (e.g., linear, sigmoid, rectified linear unit (ReLU), hyperbolic tangent, etc.) for the weight matrix $\boldsymbol{W}^{(FI)} \in \mathbb{R}^{N\times F}$. For providing a fair advantage to all the features, i.e., all features are considered as equally important, training of the feature importance layer is initialized with uniform weights $\boldsymbol{W}^{(FI)} \sim U\left(-\gamma,+\gamma\right)$, where γ will depend on the type of activation function utilized in Eq. (4.1).

The theoretical justification of the feature importance layer utilized in this study is similar to Borisov et al. [110]. The weight matrix is updated during the training process, assigning larger weights to more important features and smaller weights to less important features. To ensure that the feature importance layer correctly captures the contribution of the input features, the weights and biases of the subsequent layers must not become zero during the training process. This is maintained by utilizing the L2-norm of regularization for the weights and biases used after the feature importance layer in the training algorithm. Furthermore, to ensure sparsity of $\boldsymbol{W}^{(FI)}$, the L1-norm of regularization for $\boldsymbol{W}^{(FI)}$ is included during the training of the architecture.

4.2.2 One-dimensional Convolutional Neural Network

The CNN is a specialized neural network for detecting spatial dependencies by various types of filters (i.e., weights and bias) sliding over and convolving with the input. To address the spatial dependencies from anonymized zones in spatiotemporal forecasting, a one-dimensional CNN is utilized in this study, where the filters are slid over only across the spatial dimension (i.e., anonymized zones) of the input with spatio-temporal feature columns. To detect same kind of spatial features with a filter, parameters of a filter are shared across the zones. The output feature vector of a filter k in a one-dimensional CNN layer can be expressed as follows:

$$\boldsymbol{Z}^{(k)} = \sigma\left(\boldsymbol{X}_t * \boldsymbol{W}^{(k)} + \boldsymbol{b}^{(k)}\right) \tag{4.2}$$

where $*$ is the convolution operator, $\boldsymbol{X}_t$ is the input matrix convolved with the filter k, $\boldsymbol{Z}^{(k)} \in \mathbb{R}^N$ refers to the output feature vector for the filter, $\boldsymbol{W}^{(k)} \in \mathbb{R}^{E \times F}$ serves as the shared weight matrix of the filter with length E, and $\boldsymbol{b}^{(k)} \in \mathbb{R}^N$ is the shared bias vector of the filter. Therefore, the one-dimensional CNN layer with K filters applied to the input matrix $\boldsymbol{X}_t$ can be represented by the function $f_{Conv1D} : \mathbb{R}^{N \times F} \rightarrow \mathbb{R}^{N \times K}$.

As an example, the inputs and outputs of a one-dimensional CNN with two convolutional layers are shown in Figure 4.1. The first convolutional layer is comprised of two filters and the second convolutional layer is comprised of a single filter. The outputs of the convolutional layers are the columns of features, which are spatial patterns from the different types of input features utilized during one-dimensional convolutions with the filters. To keep the dimensions of inputs and outputs equal, zero paddings are used in the inputs of each convolutional layer. Although conventional CNN includes pooling layers, in order to prevent loss of spatial information, one-dimensional CNN in this study avoided the pooling layer following previous studies [70], [113].

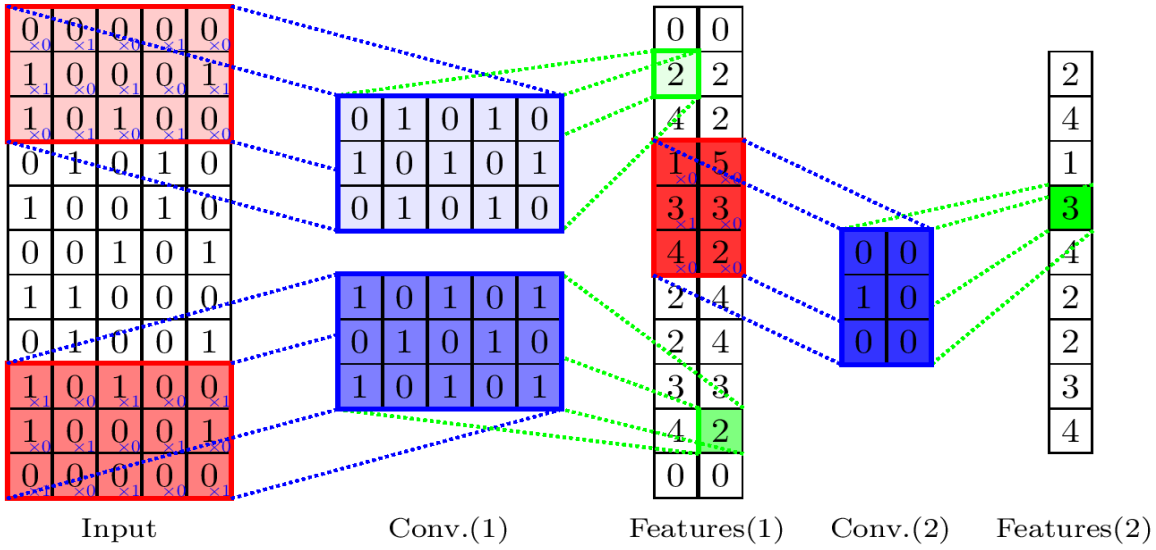


Figure 4.1. An example of one-dimensional CNN

(Source: Modified from: [42])

4.2.3 Zone-distributed Independently Recurrent Neural Network

The RNN is an exclusive architecture for detecting temporal dependencies from time-series data where the features of one time-slot are correlated with the features of the previous time-slots. In comparison to the non-recurrent connections in conventional neural networks, the RNN has recurrent connections, i.e., the outputs of the hidden layer neurons from the previous time step of a sequence are utilized with the inputs of the current time step, which can be expressed by Eq. (4.3):

$$\boldsymbol{h}_t = \sigma\left(\boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{W}\boldsymbol{h}_{t-1} + \boldsymbol{b}\right) \tag{4.3}$$

where $\boldsymbol{x}_t \in \mathbb{R}^F$ is the input vector at the current time step containing F features, $\boldsymbol{h}_t \in \mathbb{R}^H$ and $\boldsymbol{h}_{t-1} \in \mathbb{R}^H$ refer to the output vectors of size H representing hidden layer neurons of current and previous time step respectively, $\boldsymbol{U} \in \mathbb{R}^{H \times F}$ and $\boldsymbol{W} \in \mathbb{R}^{H \times H}$ serve as the weights for the input of the current step and the outputs of the previous step respectively, and $\boldsymbol{b} \in \mathbb{R}^H$ is the bias vector.

However, repeated multiplication of the recurrent hidden layer weight matrix during training results in the vanishing/exploding gradient problem in RNN that limits the storing of long-term information, which was tackled through LSTM by including additive updates in the hidden layer [45]. Despite the modification, training of LSTM can also suffer from vanishing gradient problem due to the application of non-linear activation functions [51], and can also face exploding gradient problem due to self-multiplication of the weight matrix [114]. To overcome these problems, IndRNN was proposed by replacing the entangled recurrent connections in Eq. (4.3) with an elementwise multiplication, which can be utilized to learn temporal dependencies in spatio-temporal forecasting from the independent recurrent connections, as shown in Eq. (4.4):

$$\boldsymbol{h}_t = \sigma\left(\boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{W}^{(I)} \circ \boldsymbol{h}_{t-1} + \boldsymbol{b}\right) \tag{4.4}$$

where $\boldsymbol{W}^{(I)} \in \mathbb{R}^H$ refers to the weight vector for the independently recurrent outputs of the last time step, which is regulated depending upon the activation function following the theoretical constraints provided by S. Li et al. [51] for preventing vanishing/exploding gradient problem during the training of the IndRNN.

An example of an unfolded IndRNN is shown in Figure 4.2, which has two hidden layers (each containing H neurons where h represents an individual neuron) processing an input sequence containing three time steps. In the first hidden layer, neurons independently receive the input and the corresponding outputs of the previous time step, which facilitates the detection of various temporal dependencies from the input. The subsequent hidden layer further explores the correlation among the independently learned temporal dependencies. The output vector of the current time step t is the required prediction.

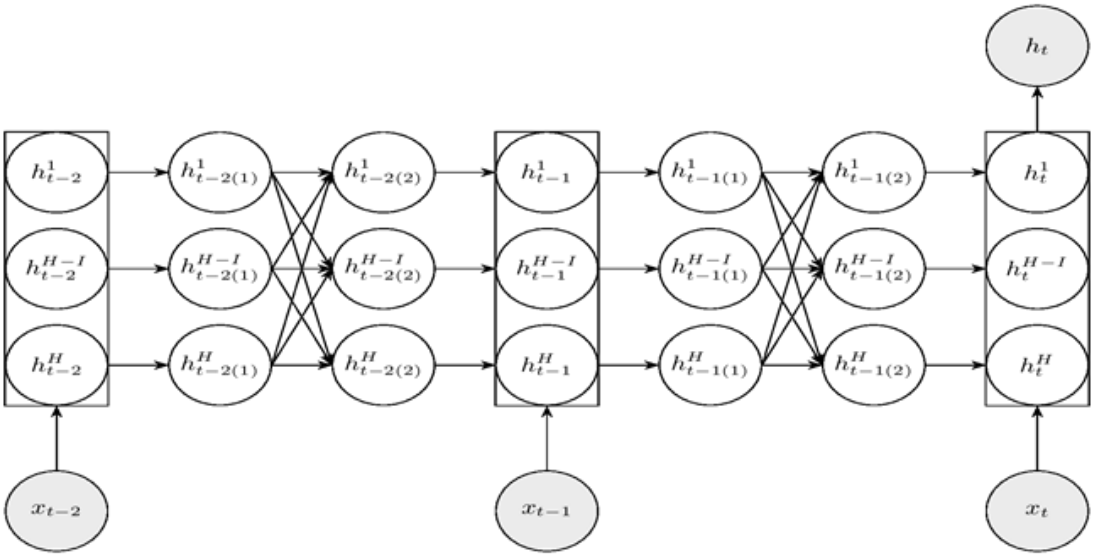


Figure 4.2. An Example of IndRNN

(Source: Modified from [115])

For detecting temporal dependencies in spatio-temporal forecasting, instead of a vector, the input of IndRNN at a time step is a matrix including the spatial dimension (i.e., zones), which requires the same IndRNN to be distributed across the zones independently with shared parameters. Therefore, the IndRNN layer is applied to the N zones of the input matrix $\boldsymbol{X}_t$ by utilizing the function $f_{ZoneDistributed(IndRNN)} : f_{IndRNN}(\boldsymbol{x}_t) \rightarrow f_{IndRNN}(\boldsymbol{X}_t)$.

4.2.4 Development of the FOCIR-Net Architecture

The workflow of the FOCIR-Net architecture is shown in Figure 4.3. All the inputs are initially spatially weighted through the feature importance layer to learn their contribution to the prediction. The weighted spatio-temporal variables are then passed through the layers of one-dimensional CNN to learn spatial dependencies, meanwhile, the weighted temporal variables together with the weighted spatiotemporal variables are passed through the stacked layers of zone-distributed IndRNN to learn temporal dependencies. Finally, the outputs of the

one-dimensional CNN and zone-distributed IndRNN are then combined with the weighted context features to make the final prediction $\boldsymbol{O}_t$. According to the problem statement, the prediction target is the demand $\boldsymbol{D}_t$ or the supply-demand gap $\boldsymbol{G}_t$, which is denoted as ground truth $\boldsymbol{A}_t$. Techniques such as concatenating and reshaping are utilized in the architecture to adapt the input with architecture. To concatenate different vectors/matrix, the concatenation function $f_{Concatenate}$ is applied (e.g., $f_{Concatenate}(\boldsymbol{P};\boldsymbol{Q}):\mathbb{R}^{N,N}\rightarrow\mathbb{R}^{N\times 2}$, where $\boldsymbol{P}\in\mathbb{R}^{N}$ and $\boldsymbol{Q}\in\mathbb{R}^{N}$ are the vectors to be concatenated). To meet the requirement of time steps for the zone-distributed IndRNN, corresponding inputs are reshaped through the reshaping function $f_{Reshape}(\boldsymbol{J};Y):\mathbb{R}^{N\times B}\rightarrow\mathbb{R}^{N\times Y\times(B/Y)}$, where $\boldsymbol{J}$ is the matrix with B features to be reshaped to Y time-steps.

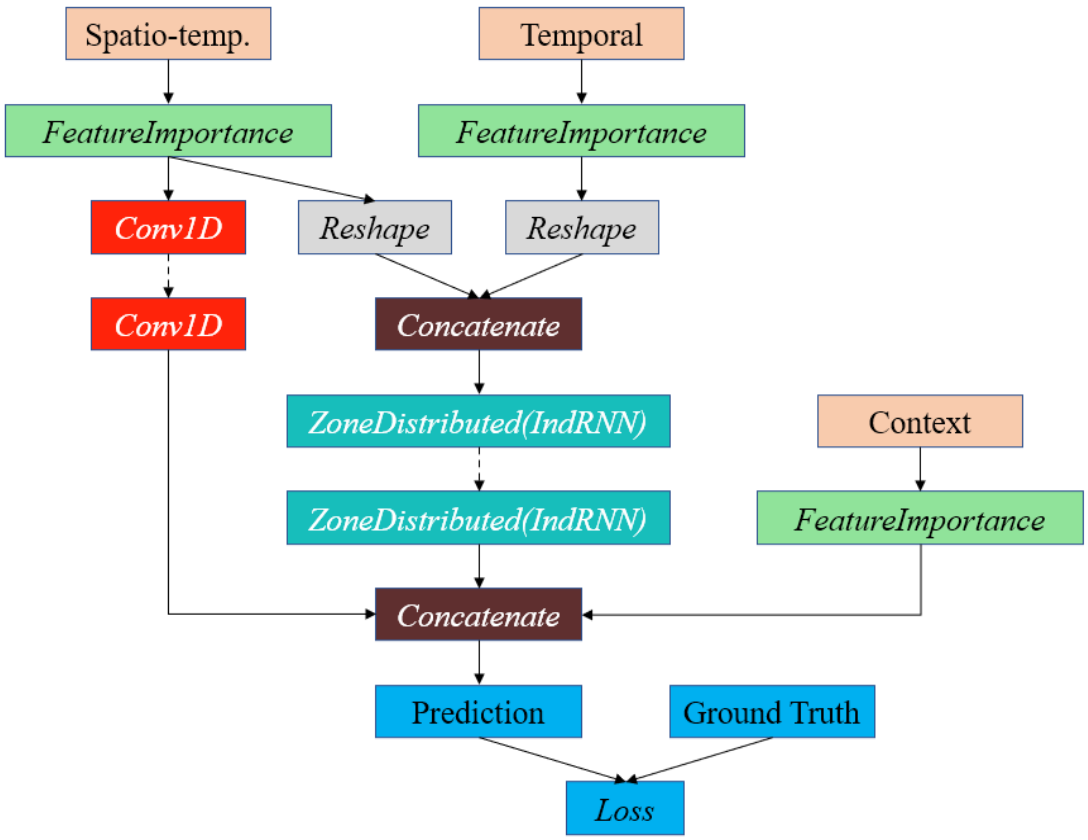


Figure 4.3. Workflow of FOCIR-Net architecture

The training of the FOCIR-Net involves minimizing the mean squared error between the predicted values and the ground truth, which is achieved through the loss function f_{Loss}. The objective of the loss function applied in the architecture including regularization terms can be expressed as Eq. (4.5):

$$\min_{W^{(A)},W^{(FI)},b} f_{Loss} = \left\| \boldsymbol{O}_t - \boldsymbol{A}_t \right\|_2^2 + \alpha \left\| \boldsymbol{W}^{(A)} \right\|_2^2 + \beta \left\| \boldsymbol{W}^{(FI)} \right\|_1 \tag{4.5}$$

where $\boldsymbol{A}_t \in \mathbb{R}^N$ is the ground truth vector, $\boldsymbol{W}^{(A)}$ refers to all parameters of the FOCIR-Net except the feature importance layer parameter $\boldsymbol{W}^{(FI)}$, and α, β are the regularization parameters. The L1- and L2-norm of regularization are utilized in accordance with the requirements of the feature importance layer, which also assists in avoiding overfitting issues.

4.3 Data Preparation, Experiments, and Model Evaluation

In this section, the proposed FOCIR-Net is tested with real-world data and the findings from the results are discussed. Furthermore, model ablation and interpretation of the proposed FOCIR-Net are also provided. Finally, the sensitivity analysis of the FOCIR-Net models is provided at the end of the section.

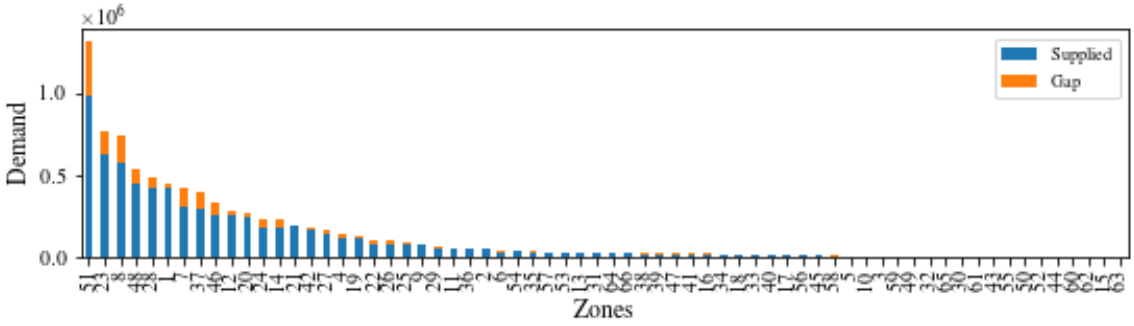
4.3.1 Data Description and Preprocessing

Publicly released ride-hailing datasets [116] from two cities of China, Beijing and Hangzhou, are selected for the experiments in this study. Didi divided each city into several square zones by applying geohashing and identified each zone with a unique ID, anonymizing the adjacency information among the zones.

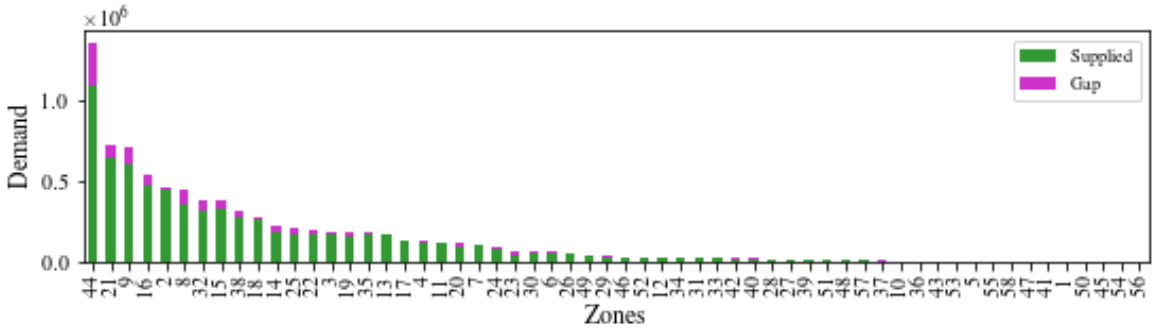
The Beijing dataset spans from 1st January 2016 to 20th January 2016 and the Hangzhou dataset spans from 23rd February 2016 to 17th March 2016. To construct time-series data for

each zone, the total timespan is divided into equal interval time-slots. Considering the small length of the datasets and focusing on short-term forecasting, each day in the datasets is therefore divided into 144 time-slots of 10 minutes interval. Furthermore, the total time-slots in a zone are split into training, validation, and testing sets for the experiments. Around 30% of the time-slots are reserved for validation and testing, and the rest of the data are used in training the models.

Both the datasets contain information around 8.5 million ride-hailing orders. The order information provides anonymized information of each order including driver ID, passenger ID, and, trip origin and destination geohashes. Besides, date and time of the orders are also available in the datasets. The unfulfilled orders are marked by a 'null' driver ID, which is useful information for calculating the supply-demand gaps in a zone for a time-slot. Furthermore, in order to find the demand and quantity supplied of each zone from the order information, the orders including 'null' driver IDs and orders excluding 'null' driver IDs are aggregated, respectively, for a timeslot. For both the datasets, around 50 percent of the time-slots are found to have zero supply-demand gaps, around 17 percent of which are due to zero demand, indicating that orders of around 33 percent time-slots are fully matched by the platform. The zone-wise distribution of total demand, including the proportions of quantity supplied and supply-demand gap, is shown in Figure 4. (a) and Figure 4. (b) for Beijing and Hangzhou respectively. In general, relatively higher demand zones tend to show a higher supply-demand gap in both datasets.



(a) Total demand distribution in Beijing



(b) Total demand distribution in Hangzhou

Figure 4.4. Total demand distribution across zones

In addition to the order information, zone-wise traffic congestion information for each time-slot is included, indicating the number of roads at different congestion levels that is aggregated to get an idea of the overall congestion of a zone for a time-slot. Furthermore, 10 minutes interval city-level weather information (i.e., weather category, PM 2.5, and temperature) and zone-wise time-invariant POI information are also provided in the datasets. The facilities in different POI classes are aggregated for the experiments

4.3.2 Model Evaluation

The proposed FOCIR-Net is compared against a set of benchmark algorithms and an ablation analysis of FOCIR-Net is conducted. For a fair comparison, all models utilized the same lookback window up to sixth previous time-slot. The settings of FOCIR-Net are decided by

tuning of the hyperparameters. The finalized settings of the hyperparameters are presented in Table 4.1.

Table 4.1. Hyperparameter settings of FOCIR-Net

	Hyperparameter settings
(a) Feature Importance Layer	Weight initialization: uniform; activation: sigmoid
(b) 1D-CNN	Layers: 2 layers; filters: 200 in 1st layer; 400 in 2nd layer; filter length: 3-11 Weight initialization: uniform; activation: ReLU
(c) Zone-distributed IndRNN	Layers: 2 layers; hidden units: 4 units per layer Weight initialization: uniform; activation: tanh/ReLU
(d) Fully Connected Layer	Weight initialization: uniform; activation: ReLU
(e) Model Training	Optimizer: Adam [99]; learning rate: 0.001; batch size: 32 Regularization: L1 = 0.001, L2 = 0.001; early stopping: patience = 100 epochs

In order to assess the performance of FOCIR-Net, GBM, XGBoost, RF, XRT, GLM, and ANN are considered, which are extensively tuned by utilizing automated machine learning frameworks. It is noteworthy to mention that, all the benchmarks utilized the same input features as FOCIR-Net.

In addition to the abovementioned models, model ablation of FOCIR-Net is also conducted by removing the model components one at a time. The following configurations are tested:

1) One-dimensional Convolution and Independently Recurrent Network (OCIR-Net): The OCIR-Net includes all model components except the feature importance layer. The input features are directly fed into the model without any spatial weighting.

2) Feature Importance Integrated One-dimensional Convolution Network (FOC-Net): The zone-distributed IndRNN is removed in this configuration. In this model, the weighted spatio-temporal features are processed through the one-dimensional CNN and the rest of the weighted features (i.e., temporal and context features) are concatenated with the outputs of the one-dimensional CNN to make the final prediction.

3) Feature Importance Integrated Independently Recurrent Network (FIR-Net): The FIR-Net excludes the one-dimensional CNN. Only temporal dependencies are considered in this model.

The performances of the models utilized in this study are evaluated with three metrics: mean absolute error (MAE), root mean squared error (RMSE), and symmetric mean absolute percentage error (sMAPE).

The proposed architecture is trained on a server with 4 Core (hyper-threaded) Xeon processor (2.30 GHz), 25 GB RAM, and a Tesla K-80 GPU. The FOCIR-Net is written in Python 3 using Keras [106] with Tensorflow [107] backend. All the benchmarks except ARIMA are implemented in H2O AutoML [108].

The performances of the FOCIR-Net and the benchmark models are reported in Table 4.2. It is evident that for all metrics, the FOCIR-Net outperforms all the benchmark models using both datasets. For demand forecasting, the FOCIR-Net has 11.32 % and 4.38 % lower RMSE than the best benchmark GBM for Beijing and Hangzhou, respectively. Furthermore, the FOCIR improves the MAE by 6.25 % and 4.21 % than the best benchmarks XGBoost for

Beijing and GBM for Hangzhou, respectively. For supply-demand gap forecasting, 9.73 % and 11.92 % improvement in RMSE and MAE, respectively, are seen for the FOCIR-Net than the best benchmarks GBM and XGBoost in Beijing, while 4.58 % improvement in MAE and marginal improvement of RMSE is found for FOCIR-Net than the best benchmark XGBoost in Hangzhou. However, the FOCIR-Net has more than 6 % improvement of sMAPE than the best benchmark XGBoost for supply-demand gap forecasting in both datasets.

Table 4.2. Performances evaluation of FOCIR-Net and benchmark models

		Metrics (Demand)				**Metrics (Supply-demand gap)**			
	Model	**MAE**	**RMSE**	**sMAPE**	**Time (s)**	**MAE**	**RMSE**	**sMAPE**	**Time (s)**
(a) Beijing	FOCIR-Net	6.45	16.77	0.1867	229	3.34	14.56	0.2288	555
	XGBoost	6.88	19.07	0.2009	3.68	3.70	16.74	0.2952	11.89
	GBM	7.00	18.91	0.2186	3.99	3.77	16.53	0.3053	3.43
	XRT	7.04	19.43	0.1994	78.40	3.91	17.07	0.3045	31.03
	RF	7.23	22.18	0.2006	79.45	3.90	16.60	0.3044	37.32
	GLM	7.65	19.78	0.2537	0.44	4.61	16.80	0.3998	1.01
	ANN	14.48	25.57	0.4250	33.21	5.34	19.09	0.4671	114.74
(b) Hangzhou	FOCIR-Net	6.82	16.34	0.1626	237	2.92	13.45	0.2257	181
	XGBoost	7.25	17.35	0.1770	17.18	3.06	13.46	0.2873	14
	GBM	7.12	17.09	0.1761	10.69	3.06	13.52	0.3018	4.24
	XRT	7.19	17.43	0.1734	106.97	3.12	13.64	0.2898	59.59
	RF	7.18	17.28	0.1730	103.09	3.14	13.80	0.2896	58.76
	GLM	7.70	18.05	0.2070	0.42	3.44	14.55	0.3222	0.82
	ANN	7.80	18.12	0.2137	7.67	3.41	14.07	0.3426	7.85

Table 4.3 presents the performance evaluation of the models utilized in the ablation analysis of the FOCIR-Net. For both datasets, improvements around 1-2 % in the MAE and RMSE of the FOCIR-Net are observed with respect to the corresponding values of MAE and RMSE in OCIR-Net and FOC-Net, respectively, which indicates that the feature importance layer and zone-distributed IndRNN adds to the performance of the FOCIR-Net. Most importantly, the large drop in performance with a huge increase in training time in the FIR-Net indicates that the processing of the spatio-temporal variables through the one-dimensional CNN plays the most important role in the FOCIR-Net.

Table 4.3. Ablation analysis of FOCIR-Net

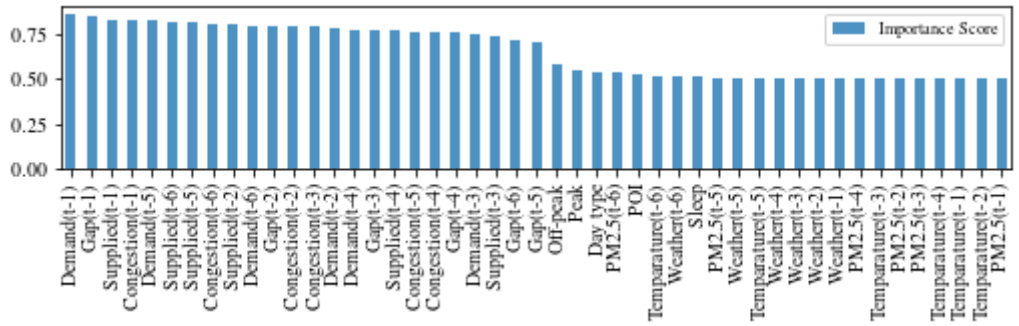
		Metrics (Demand)				**Metrics (Supply-demand gap)**			
	Model	**MAE**	**RMSE**	**sMAPE**	**Time (s)**	**MAE**	**RMSE**	**sMAPE**	**Time (s)**
(a) Beijing	OCIR-Net	6.71	17.19	0.1942	152	3.38	14.67	0.2351	691
	FOC-Net	6.49	16.96	0.1958	229	3.42	14.98	0.2372	260
	FIR-Net	19.61	51.00	0.3116	3150	7.44	33.13	0.3057	1982
(b) Hangzhou	OCIR-Net	7.02	16.68	0.1718	394	2.93	13.59	0.2234	326
	FOC-Net	7.04	16.66	0.1693	312	3.08	13.91	0.2298	236
	FIR-Net	7.71	18.25	0.1928	777	3.61	15.25	0.2583	726

4.3.3 Model Interpretations

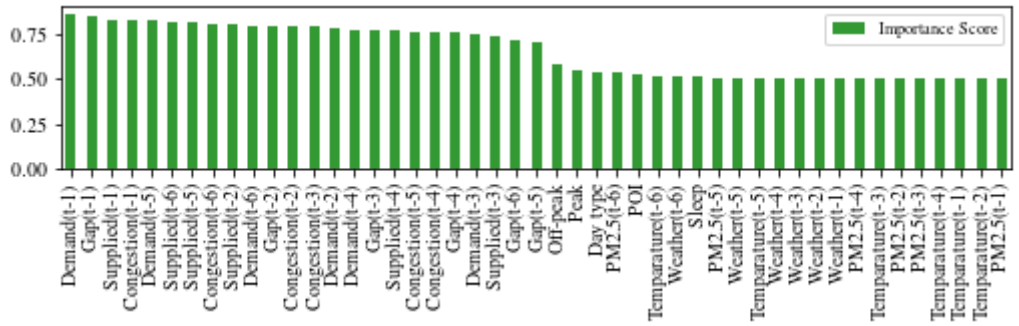
In this study, the weights utilized by the feature importance layer in the FOCIR-Net are processed through a sigmoid function, which bounds the learned weights between 0 and 1. This is useful to interpret the contributions of the input features in the FOCIR-Net. In order to separately explain the contribution of the features temporally and spatially, the outputs of the

feature importance layer are averaged spatially and temporally, respectively. To avoid repetitions, only the results for Hangzhou data is presented here.

Figure 4. (a) and Figure 4. (b) presents the feature importance rankings of the spatially averaged features for demand and supply-demand gap forecasting, respectively. It is intriguing to find that the historical values of the demand, quantity supplied, and congestion have a major contribution to the prediction. The least contribution among the spatio-temporal features is seen for the historical values of the supply-demand gap, which is mainly because of irregular patterns in the historical supply-demand gap. It is also found that almost all the context features (except the sleep hour in the supply-demand gap forecasting) are found to have more contribution towards prediction while comparing with the temporal features. Among the temporal features, the temperature and the PM2.5 for the interval (t-5) and (t-6) have relatively higher importance than the other temporal features.



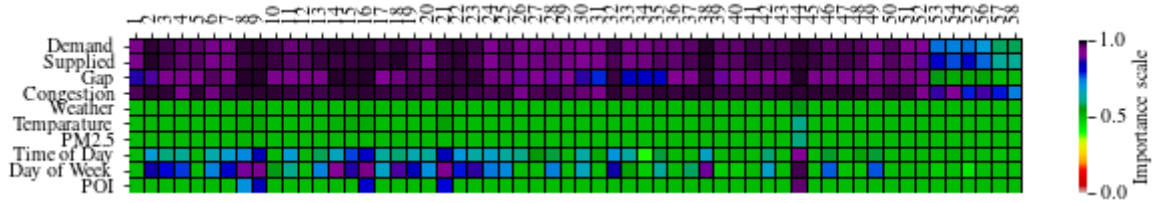
(a) Feature importance ranking for demand forecasting



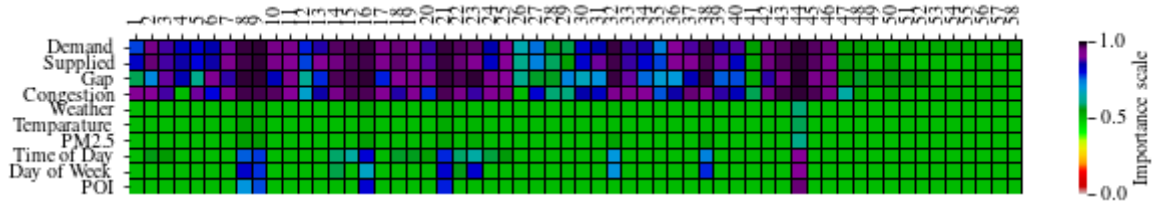
(b) Feature importance ranking for supply-demand gap forecasting

Figure 4.5. Spatially averaged feature importance

The temporally averaged features are presented in the heatmaps in Figure 4. (a) and Figure 4. (b). It is found that importance of the spatio-temporal features and the context features are not uniform across the zones, while importances of the temporal features are mostly uniform across the zones except for the zone-44, which has the highest total of demand and supply-demand gap. In general, it is found that the features of the higher demand (supply-demand gap) zones are more important to the FOCIR-Net for prediction than that of the lower demand (supply-demand gap) zones.



(a) Feature importance heatmap for demand forecasting

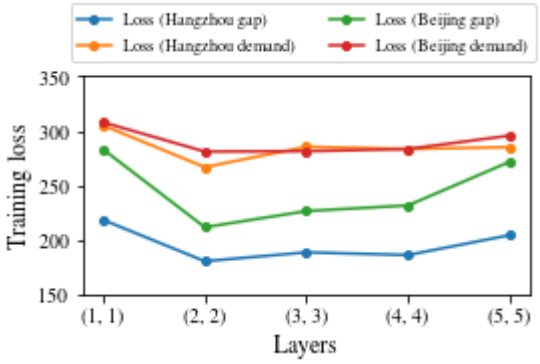


(b) Feature importance heatmap for supply-demand gap forecasting

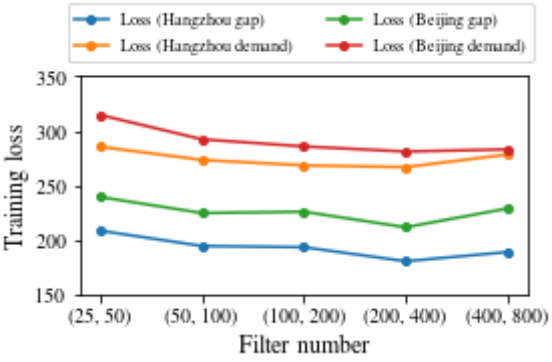
Figure 4.6. Temporally averaged feature importance

4.3.4 Sensitivity Analysis

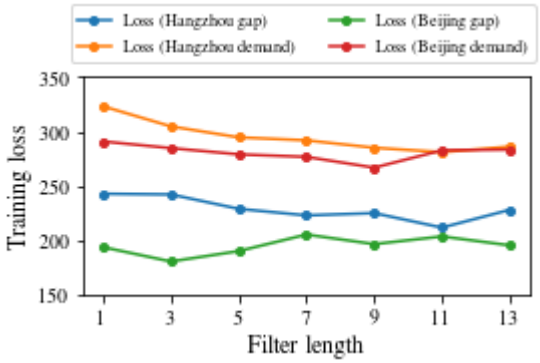
A sensitivity analysis of the FOCIR-Net is conducted in terms of four hyperparameters: number of one-dimensional CNN and zone-distributed IndRNN layers, number of filters in one-dimensional CNN, filter length of one-dimensional CNN, and the number of hidden units in zone-distributed IndRNN. In order to get the best performance for every hyperparameter configuration, all the experiments on hyperparameter tuning are conducted with early stopping.



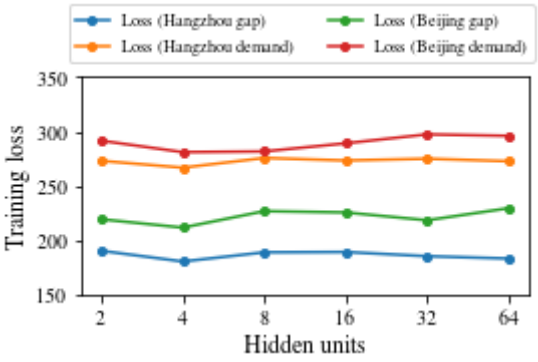
(a) Number of 1D-CNN and IndRNN layers



(b) Number of 1D-CNN filters



(c) 1D-CNN filter length



(d) IndRNN hidden layer units

Figure 4.7. Sensitivity analysis of FOCIR-Net hyperparameters

The variation of the training loss (i.e., mean squared error) for forecasting demand and supply-demand gap in both datasets with respect to the number of layers is shown in Figure 4. (a). The model losses reach the lowest for 2 layers and then rise again with an increased number of layers. This result is not unintuitive since it is theoretically proven that neural networks with 2 hidden layers can approximate any continuous function [109]. The performance of FOCIR-Net with varying number of filters in the one-dimensional CNN is shown in Figure 4. (b). It is evident that training losses in both datasets gradually decreases and achieves the minimum for a combination of 200 filters in the first layer and 400 filters in the second layer, while filter numbers greater than the best combination show deterioration in performance due to overfitting. Different effects of filter length on the training loss are seen

in Figure 4. (c). The model losses follow a decreasing trend up to the filter length of 11 for both demand and supply-demand gap in the Beijing dataset, whereas filter lengths of 3 and 9, respectively, reach the minimum loss for demand and supply-demand gap in the Hangzhou dataset. The varied outcomes of the most suitable filter lengths are not unexpected because of the underlying spatial dependencies, which may vary depending on the forecasting tasks and datasets. Comparing with other hyperparameters, variations of the zone-distributed IndRNN hidden units show very small changes in the training losses for both datasets, showing a common best performance achieved by 4 hidden units in Figure 4. (d). The small variations for zone-distributed IndRNN hidden units are most likely due to the fact that most of the learnings in FOCIR-Net are done by the one-dimensional CNN.

CHAPTER 5: DEVELOPING SPATIO-TEMPORAL MULTI-TASK LEARNING ARCHITECTURE FOR RIDE-HAILING SYSTEM

This chapter discusses the development of the spatio-temporal multi-task learning architecture for simultaneous forecasting of multiple spatio-temporal tasks in ride-hailing system. A detailed description the methodology and mathematical explanation of the proposed GESME-Net architecture is given in this chapter, followed by evaluation the proposed architecture with real-world datasets of Didi and comparison of the result with the benchmark models. Furthermore, model ablation and interpretation of the proposed Finally, a sensitivity analysis of the spatio-temporal deep learning models considered in this chapter are provided.

5.1 Preliminaries and Problem Statement

This section presents a description of the variables utilized in this study and the problem of spatio-temporal multi-task learning in a ride-hailing system. Spatio-temporal forecasting in ride-hailing system involve time-series data, therefore, historical values of the variables that evolve with time are important indicators for predicting targets. Additionally, POI, time of day, and day of week also have effects on the spatio-temporal forecasting.

Space-time partitioning: For aggregating the variables, the study area is divided into N non-overlapping uniformly sized zones $Z = \{1, 2, 3, ..., N\}$ and total time is divided into T time-slots $I = \{1, 2, 3, ..., T\}$ of m minutes interval. The rest of the variables are explained based on this space-time partitioning.

Spatio-temporal variables: Spatio-temporal variables vary simultaneously in the spatial and temporal dimension. The following types of the spatio-temporal variables are utilized in this study:

(1) Demand: The ride-hailing demand at all zones during the time-slot $t \in I$ is represented by the vector $\boldsymbol{D}_t \in \mathbb{R}^N$, where $D_{t,z}$ is the total number of successful ride-hailing requests emerging from zone $z \in Z$, and $D_{t,z} \in [0, +\infty)$.

(2) Original demand: The total number of ride-hailing requests from a zone in a time interval is referred to as the original demand, which includes both successfully matched and unanswered ride-hailing requests. The original demand of all zones during the time-slot t is placed in the vector $\boldsymbol{OD}_t \in \mathbb{R}^N$. The demand at zone z during the time-slot t is denoted as $OD_{t,z}$, where $OD_{t,z} \geq D_{t,z}$.

(3) Supply-demand gap: The total number of unanswered ride-hailing requests from a spatial zone in a time-slot is termed as the supply-demand gap. The supply-demand gap of all zones during the time-slot t is expressed as the vector $\boldsymbol{G}_t \in \mathbb{R}^N$. The supply-demand gap of a zone z during the time-slot t is denoted by $G_{t,z}$, where $0 \leq G_{t,z} \leq OD_{t,z}$.

(4) Traffic congestion: The traffic congestion of all zones is denoted by the vector $\boldsymbol{TC}_t \in \mathbb{R}^N$, where $TC_{t,z}$ represents the total number of congested roads belonging to a zone z during the time-slot t.

(5) Speed: The average speed of the floating ride-hailing cars of a spatial zone in a time-slot is termed as the speed in that spatial zone. The speed at all zones during the time-slot t is denoted by the vector $\boldsymbol{S}_t \in \mathbb{R}^N$. The speed at a zone z during the time-slot t is denoted by $S_{t,z}$.

(6) Accessibility: The accessibility of a spatial zone in a time-slot is measured by the number of ride-hailing cars with passengers crossing that spatial zone. The accessibility of all zones in the time-slot t is expressed by the vector $\boldsymbol{M}_t \in \mathbb{R}^N$. The accessibility of a zone z during the time-slot t is denoted by $M_{t,z}$.

Weather variables: The weather variables include the meteorological features that vary randomly across time, but not space. For maintaining consistency of input dimensions in the proposed architectures, these variables may have to be repeated across the zones by utilizing the repeating function $f_{RZ}(\cdot;N):\mathbb{R}^{1\times M}\to\mathbb{R}^{N\times M}$, where M represents the number of feature categories. The following types of weather variables are included in this study:

(1) Categorical weather variables: For the time-slot t, the weather conditions are the categorical weather variables, denoted by the row vector $\boldsymbol{wc}_t\in\mathbb{R}^{1\times C}$ consisting of C weather categories (e.g., sunny, rainy, snowy, etc.), where each weather category is encoded by one-hot encoding, i.e., $wc_{c,t}\in\{0,1\}$.

(2) Continuous weather variables:

For the time-slot t, a continuous weather variable is expressed by the vector $\boldsymbol{wt}_t\in\mathbb{R}$. The continuous weather variables utilized in this study are temperature, particulate matter, dew point, humidity, cloud cover, wind speed, and visibility.

Context variables: The context variables are either periodic or fixed across time. For spatio-temporal forecasting, these variables are either repeated across the zones by applying the repeating function f_{RZ} or repeated across the time-slots by utilizing the repeating function $f_{RT}(\cdot;T):\mathbb{R}^{N}\to\mathbb{R}^{N\times 1\times T}$. These are as follows:

(1) Temporal context: Following [4], exploratory data analysis of the trends in demand and supply-demand gap revealed two types of periodic contexts: time-of-day and day-of-peak. A time-slot t of a day belongs to one of the three 8-hour time-of-day intervals: sleep (first 8-hour), peak (mid-8-hour), and off-peak (last 8-hour), denoted by the row vector $\boldsymbol{cd}_t\in\mathbb{R}^{1\times 3}$, where each interval i is one-hot encoded, i.e., $cd_{i,t}\in\{0,1\}$. Furthermore, a time-slot t falls into one of the day-of-week categories: weekday or weekend, represented by the vector $\boldsymbol{cw}_t\in\mathbb{R}$, where $cw_t\in\{0,1\}$. The temporal context vectors $\boldsymbol{cd}_t$ and $\boldsymbol{cw}_t$ are repeated across

the zones to form the time-of-day matrix $\boldsymbol{CD}_t \in \mathbb{R}^{N\times 3}$ and the day-of-week vector $\boldsymbol{CD}_t \in \mathbb{R}^{N}$ respectively.

(2) Spatial context: The spatial context refers to the number of POIs across the zones, denoted by the vector $\boldsymbol{cp}_z \in \mathbb{R}^{N}$, which is fixed across time. Therefore, it is repeated across the time-slots with the repeating function f_{RT} to form the spatial context vector for the time-slot t, represented by the vector $\boldsymbol{CP}_t \in \mathbb{R}^{N}$.

With the abovementioned definition of the variables, the spatio-temporal forecasting problem can be formulated as follows:

Problem Statement: For n spatio-temporal tasks in a ride-hailing system, given, the historical data of spatio-temporal variables and weather variables up to b th previous time-slot starting from $t-1$, and known data of context variables at the time-slot t, it is required to predict the ground truth vector $\boldsymbol{A}_t$ at the time-slot t for n spatio-temporal forecasting tasks simultaneously.

5.2 Methodology

This section first provides a brief description of the feature weighting layer and the mixture of experts, which are the core methods for dealing with multi-task learning in the proposed architecture. Furthermore, developments of the gated mixture of experts for the different components, i.e., CNN, RNN, and Conv-RNN are also explained. Finally, the proposed GESME-Net architecture is explained.

5.2.1 Feature Weighting Layer

Multi-task learning methods involve capturing complex interaction among features through multiple intermediate layers between inputs and outputs in such a way that a joint

representation for all task is learnt. However, less important input features may hinder the learning process in multi-task learning. To address this issue, feature weighting based on a one-to-one linear layer [110]–[112], usually utilized after the input layer, is implemented in the proposed architecture for integrating feature importance in multi-task learning. The main idea of a one-to-one linear layer is similar to a fully connected layer except that a neuron in a one-to-one linear layer is connected with only one input rather than all inputs. This study applies a similar idea of feature weighting, however, an input-agnostic feature weighting layer, i.e., adaptable with feature of any dimension, for spatio-temporal forecasting is developed, which can be expressed by the following function in Eq. (5.1):

$$f_{Weighting}\left(\boldsymbol{X}_t\right) = \boldsymbol{X}_t \odot \sigma(\boldsymbol{W}^{(FI)}) \tag{5.1}$$

The operator $\odot$ in Eq. (5.1) indicates Hadamard product, i.e., elementwise multiplication, between any input $\boldsymbol{X}_t$ for the time-slot t and the outputs of the activation function σ (e.g., linear, sigmoid, rectified linear unit (ReLU), hyperbolic tangent, etc.) for the weights $\boldsymbol{W}^{(FI)}$. For providing a fair advantage to all the features, i.e., all features are considered as equally important, training of the feature weighting layer is initialized with uniform weights $\boldsymbol{W}^{(FI)} \sim U\left(-\gamma, +\gamma\right)$, where γ will depend on the type of activation function utilized in Eq. (5.1).

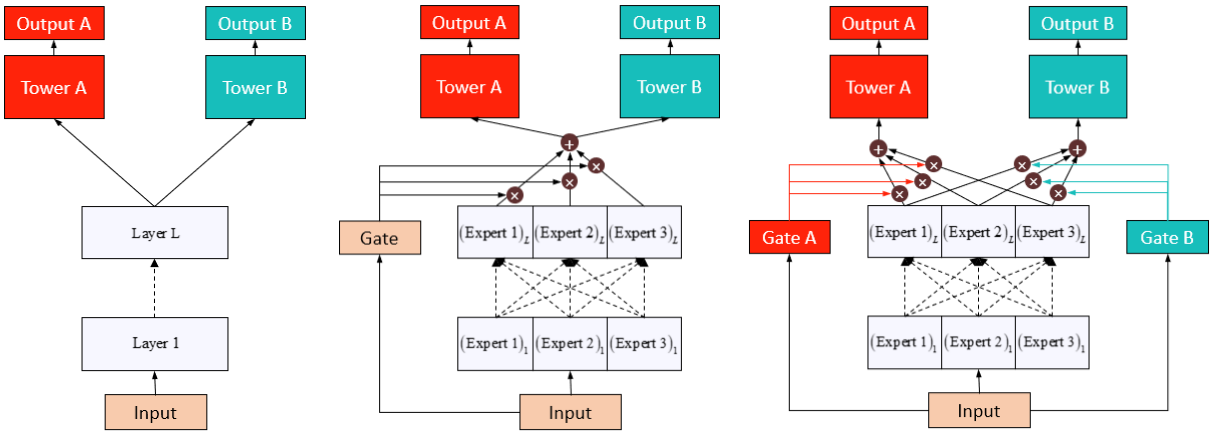
The theoretical justification of the feature weighting layer utilized in this study is similar to Borisov et al. [110] except that a linear activation function is utilized in Eq. (5.1). The weights are updated during the training process, assigning larger weights to more important features and smaller weights to less important features. To ensure that the feature weighting layer correctly captures the contribution of the input features, the weights and biases of the subsequent layers must not become zero during the training process. This is maintained by utilizing the L2-norm of regularization for the weights and biases used after the feature importance layer in the training algorithm. Furthermore, to ensure sparsity of $\boldsymbol{W}^{(FI)}$, the L1-norm of regularization for $\boldsymbol{W}^{(FI)}$ is included in the training algorithm.

5.2.2 *Mixture of Experts*

The mixture of experts was initially developed as an ensemble method [117] and later utilized as stacked layers in deep learning models [118]. Instead of a single neural network, a mixture of experts contains subnetworks called experts and a gating network that learns the probabilities of the experts. A mixture of experts model can be expressed by the following function:

$$f_{ME}(\boldsymbol{X}_t) = \sum_{i=1}^{m} f_{gate}(\boldsymbol{X}_t)_i \times f_{\text{expert}}^{i}(\boldsymbol{X}_t) \qquad (5.2)$$

where the functions f_{gate} and f_{expert} represents the gating and expert network respectively, m represents the number of expert networks utilized, and $\sum_{i=1}^{m} f_{gate}(\boldsymbol{X}_t)_i = 1$.



(a) Shared-bottom *(b) Shared-gated ME* *(c) Multi-gated ME*

Figure 5.1. Multi-task learning models

(Source: Reproduced from [57])

The concept of multi-task learning through gated mixture of experts [57] is developed based on the commonly used shared-bottom neural network (SBNN) [119] for multi-task learning, which is shown in Figure 5.1 (a). In the SBNN, the bottom layers utilize shared parameters for all tasks and then task-specific layers (i.e., tower) are introduced following the shared layers. The mathematical formulation of a shared-bottom multi-task learning model for predicting set of n tasks $P = \{1, 2, 3, ..., n\}$ can be expressed as follows:

$$\boldsymbol{Y}_t^p = h^p(f_{shared}(\boldsymbol{X}_t)) \tag{5.3}$$

where, $\boldsymbol{Y}_t^p$ is the represents the output vector of the task $p \in P$ at the time-slot t, f_{shared} and h^p represents shared bottom layers and the task-specific layers respectively.

However, SBNN are only found to be useful in multi-task learning settings with similar tasks [120]. Therefore, instead of the shared bottom layers, J. Ma et al. [57] utilized gated mixture of experts, as shown in Figure 5.1 (b) and Figure 5.1 (c). The gating networks produce task-specific softmax probabilities for each expert by processing the input features, providing diversity and flexibility in learning different tasks through the experts. The outputs of the ensembled experts are finally forwarded to the task-specific layers. Therefore, by modifying Eq. (5.2)-(5.3), the gated mixture of experts for multi-task learning is expressed through the Eq. (5.4)-(5.5).

$$\boldsymbol{Y}_t^p = h^p(f_{ME}^p(\boldsymbol{X}_t)) \tag{5.4}$$

$$f_{ME}^p(\boldsymbol{X}_t) = \sum_{i=1}^{m} f_{gate}^p(\boldsymbol{X}_t)_i \times f_{\text{expert}}^i(\boldsymbol{X}_t) \tag{5.5}$$

Although such ensemble of experts is proven to learn task relationships in multi-task learning, the experts utilized are feedforward neural networks that are unable to learn spatio-temporal dependencies. Therefore, gated mixture of experts based on CNN, RNN, and Conv-RNN are developed in this study for learning spatio-temporal dependencies in multi-task learning.

5.2.2.1 Gated Convolutional Mixture of Experts

The CNN is a specialized neural network for detecting spatial dependencies by various types of filters (i.e., weights and bias) sliding over and convolving with the input. However, spatial dependencies in spatio-temporal forecasting not only depends on neighboring zones but also distant zones (e.g., functional similarity, transportation connectivity) [9]. Furthermore, the spatial adjacency information is sometimes anonymized in the ride-hailing datasets due to confidentiality reasons. Therefore, a one-dimensional CNN is utilized in this study for multi-task learning, based on the findings of this study in the previous two chapters. The one-dimensional CNN filters are slid over only across the flattened spatial dimension of the input with spatio-temporal feature columns. To detect same kind of spatial features with a filter, parameters of a filter are shared across the zones. The output feature vector of a filter k in a one-dimensional CNN layer can be expressed as follows:

$$\boldsymbol{Z}^{(k)} = \sigma\left(\boldsymbol{X}_t * \boldsymbol{W}^{(k)} + \boldsymbol{b}^{(k)}\right) \tag{5.6}$$

where $*$ is one-dimensional convolution operator, $\boldsymbol{X}_t$ is the input to be convolved with the filter k, $\boldsymbol{Z}^{(k)}$ refers to the output feature vector for the filter, $\boldsymbol{W}^{(k)}$ serves as the shared weight matrix of the filter, and $\boldsymbol{b}^{(k)}$ is the shared bias vector of the filter. Therefore, the one-dimensional CNN layer with K filters applied to the input $\boldsymbol{X}_t$ with N zones and F features can be represented by the function $f_{Conv1D} : \mathbb{R}^{N \times F} \rightarrow \mathbb{R}^{N \times K}$.

We developed gated convolutional mixture of experts for detecting special dependencies in multi-task learning by utilizing one-dimensional convolution instead of feedforward neural networks as the experts in Eq. (5.5), which is expressed as follows:

$$f^{p}_{Conv-ME}(\boldsymbol{X}_t) = \sum_{i=1}^{m} f^{p}_{gate}(\boldsymbol{X}_t)_i \times f^{i}_{Conv1D}(\boldsymbol{X}_t) \tag{5.7}$$

5.2.2.2 Gated Recurrent Mixture of Experts

The RNN is an exclusive architecture for detecting temporal dependencies from time-series data where the features of one time-slot are correlated with the features of the previous time-slots. In comparison to the non-recurrent connections in conventional neural networks, the RNN has recurrent connections, i.e., the outputs of the hidden layer neurons from the previous time step of a sequence are utilized with the inputs of the current time step, which is already expressed in Eq. (5.8):

$$\boldsymbol{h}_t = \sigma\left(\boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{W}\boldsymbol{h}_{t-1} + \boldsymbol{b}\right) \tag{5.8}$$

where $\boldsymbol{x}_t \in \mathbb{R}^F$ is the input vector at the current time step containing F features, $\boldsymbol{h}_t \in \mathbb{R}^H$ and $\boldsymbol{h}_{t-1} \in \mathbb{R}^H$ refer to the output vectors of size H representing hidden layer neurons of current and previous time step respectively, $\boldsymbol{U} \in \mathbb{R}^{H \times F}$ and serve as the weights for the input of the current step and the outputs of the previous step respectively, and $\boldsymbol{b} \in \mathbb{R}^H$ is the bias vector.

However, repeated multiplication of the recurrent hidden layer weight matrix during training results in the vanishing/exploding gradient problem in RNN that limits the storing of long-term information, which can be tackled through LSTM [45] and GRU [121] by including additive updates in the hidden layer. Although LSTM and GRU are found to be similar in terms of performance [122], training GRU requires lesser time, which is an advantage for learning large number of tasks with multi-task learning. Therefore, GRU is chosen for developing gated recurrent mixture of experts in this study. The GRU cell, as shown in Figure 5.2, modifies the recurrent structure in Eq. (4.3) through Eq. (5.9)-(5.12):

$$\boldsymbol{z}_t = \sigma\left(\boldsymbol{U}^{(z)}\boldsymbol{x}_t + \boldsymbol{W}^{(z)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(z)}\right) \tag{5.9}$$

$$\boldsymbol{r}_t = \sigma\left(\boldsymbol{U}^{(r)}\boldsymbol{x}_t + \boldsymbol{W}^{(r)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(r)}\right) \tag{5.10}$$

$$\tilde{\boldsymbol{h}}_t = \tanh\left(\boldsymbol{U}^{(h)}\boldsymbol{x}_t + \boldsymbol{r}_t \odot \boldsymbol{W}^{(h)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(h)}\right) \tag{5.11}$$

$$\boldsymbol{h}_t = \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t + (1 - \boldsymbol{z}_t)\boldsymbol{h}_{t-1} \tag{5.12}$$

where $\boldsymbol{z}_t$, $\boldsymbol{r}_t$, and $\tilde{\boldsymbol{h}}_t$ are update gate, reset gate, and new states respectively, containing corresponding input weights, recurrent weights, and bias vector. The recurrent layer with gated recurrent units can be denoted by the function $f_{GRU} : \mathbb{R}^F \rightarrow \mathbb{R}^H$.

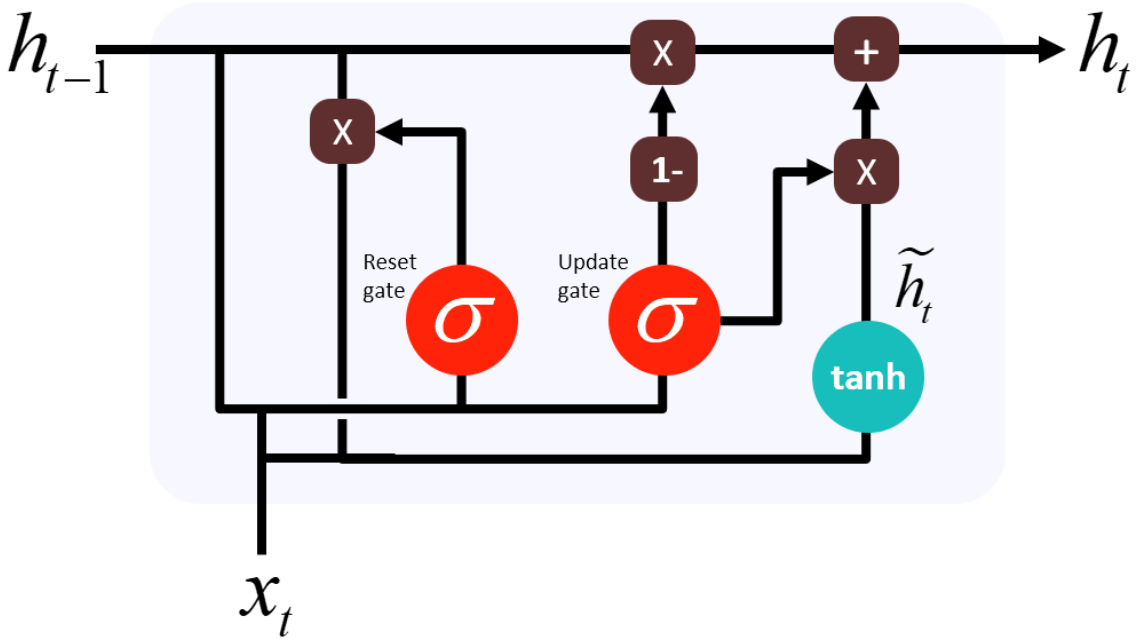


Figure 5.2. A GRU cell

(Source: Modified from [123])

For detecting temporal dependencies in spatio-temporal features, instead of a vector, the input of GRU at a time step is a matrix $\boldsymbol{X}_t \in \mathbb{R}^{N \times F}$ including N zones, which requires the same GRU layer to be distributed across the zones independently with shared parameters. Therefore, the GRU layer is applied to the N zones of the input matrix $\boldsymbol{X}_t$ with the function $f_{ZoneDist(GRU)} : f_{GRU}(x_t) \rightarrow f_{GRU}(\boldsymbol{X}_t)$.

The gated recurrent mixture of experts for learning temporal dependencies in multi-task learning can be formulated by replacing the non-recurrent gates and experts in Eq. (5.5) with recurrent gates and recurrent experts as follows:

$$f^{p}_{GRU-ME}(\boldsymbol{x}_t) = \sum_{i=1}^{m} f^{p}_{GRUgate}(\boldsymbol{x}_t)_i \times f^{i}_{GRU}(\boldsymbol{x}_t) \tag{5.13}$$

$$f^{p}_{ZoneDist(GRU)-ME}(\boldsymbol{X}_t) = \sum_{i=1}^{m} f^{p}_{ZoneDist(GRU)gate}(\boldsymbol{X}_t)_i \times f^{i}_{ZoneDist(GRU)}(\boldsymbol{X}_t) \tag{5.14}$$

where the function f^{p}_{GRU-ME} and $f^{p}_{ZoneDist(GRU)-ME}$ represents gated mixture of experts and zone-distributed gated mixture of experts respectively with $f^{p}_{GRUgate}$ and $f^{p}_{ZoneDist(GRU)gate}$ are the corresponding recurrent gating layers for producing softmax probabilities following a GRU layer.

5.2.2.3 Gated Convolutional Recurrent Mixture of Experts

Convolutional recurrent layers are exclusively utilized to detect spatial and temporal dependencies simultaneously from spatio-temporal features. To prevent vanishing/exploding gradient problems in convolutional recurrent layers, convolutional LSTM [47] and gated convolutional recurrent unit [94] have been developed. However, the use of LSTM/GRU in the convolutional recurrent layer makes it computationally expensive due to large number of output units [48] and does not provide better model performance other than that they only ease the training process [49]. Rather, a convolutional recurrent layer with a vanilla RNN (ConvRNN) can easily tackle the vanishing/exploding gradient problem by utilizing a linear activation function, while being computationally cheaper. A ConvRNN can be formulated by modifying the cell structure of the vanilla RNN in Eq. (5.8) to take spatio-temporal features as inputs and replacing the matrix multiplication with a convolution operator. Therefore, a one-dimensional ConvRNN is implemented in this study, which is expressed as follows:

$$\boldsymbol{H}_t = ReLU\left(\boldsymbol{U}^{(c)} * \boldsymbol{X}_t + \boldsymbol{W}^{(c)} * \boldsymbol{H}_{t-1} + \boldsymbol{b}^{(c)}\right) \tag{5.15}$$

where $\boldsymbol{X}_t \in \mathbb{R}^{N \times F}$ is the input at the current time step containing N zones and F features, $\boldsymbol{H}_t \in \mathbb{R}^{N \times K}$ and $\boldsymbol{H}_{t-1} \in \mathbb{R}^{N \times K}$ refer to the output of current and previous time step respectively, $\boldsymbol{U}^{(c)}$ and $\boldsymbol{W}^{(c)}$ serve as the filters for the input of the current step and the outputs of the previous step respectively, and $\boldsymbol{b}^{(c)} \in \mathbb{R}^{N \times K}$ is the bias. Therefore, a convolutional recurrent layer processing B timesteps can be denoted as the function $f_{ConvRNN} : \mathbb{R}^{N \times F \times B} \rightarrow \mathbb{R}^{N \times K \times B}$.

The gated convolutional recurrent mixture of experts for multi-task learning is developed by utilizing the ConvRNN as follows:

$$f^{p}_{ConvRNN-ME}(\boldsymbol{X}_t) = \sum_{i=1}^{m} f^{p}_{ConvRNNgate}(\boldsymbol{X}_t)_i \times f^{i}_{ConvRNN}(\boldsymbol{X}_t) \qquad (5.16)$$

where the function $f^{p}_{ConvRNN-ME}$ represents gated mixture of experts and $f^{p}_{ConvRNNgate}$ is the recurrent gating layer for producing softmax probabilities from the outputs of a ConvRNN layer.

5.2.3 Development of the GESME-Net Architecture

The workflow of the GESME-Net architecture is shown in Figure 5.3. All the inputs are initially weighted through the feature weighting layer to adjust their importance in multi-task learning. The weighted spatio-temporal variables are then passed through the layers of gated convolutional recurrent mixture of experts to learn task-specific spatial and temporal dependencies simultaneously from the spatio-temporal variables. Furthermore, to independently learn spatial and temporal dependencies from the spatio-temporal features, they are passed through the gated convolutional mixture of experts together with the spatial context features to learn the task-specific spatial dependencies and through the zone-distributed gated recurrent mixture of experts to learn the task-specific temporal dependencies. The weighted weather features are separately passed through a gated recurrent

mixture of experts to learn the task-specific dependencies on the weather features. Finally, the task-specific outputs of the different mixture of experts are combined with the weighted temporal context features and passed through a fully connected layer (i.e., dense) to make the final prediction $\boldsymbol{O}_t$ in the tower layer. According to the problem statement, the prediction target is denoted as the ground truth $\boldsymbol{A}_t$. Techniques such as concatenating, reshaping, and permuting dimensions are utilized in the architecture to adapt the input requirement of the architecture. To concatenate different vectors/matrix, the concatenation function $f_{Concatenate}$ is applied (e.g., $f_{Concatenate}(\boldsymbol{P};\boldsymbol{Q}):\mathbb{R}^{N,N}\rightarrow\mathbb{R}^{N\times 2}$, where $\boldsymbol{P}\in\mathbb{R}^{N}$ and $\boldsymbol{Q}\in\mathbb{R}^{N}$ are the vectors to be concatenated). The function $f_{Permute}$ is applied to permute the dimensions of a tensor (e.g., $f_{Permute}(\boldsymbol{V};(2,3,1)):\mathbb{R}^{C\times D\times E}\rightarrow\mathbb{R}^{D\times E\times C}$, where $\boldsymbol{V}\in\mathbb{R}^{C\times D\times E}$ is the tensor to be permuted with the provided permuted ordering of the dimensions). To meet the requirement of time steps for the zone-distributed recurrent mixture of experts, corresponding inputs are reshaped through the reshaping function $f_{Reshape}(\boldsymbol{J};B):\mathbb{R}^{N\times A}\rightarrow\mathbb{R}^{N\times B\times(A/B)}$, where $\boldsymbol{J}$ is the matrix with A features to be reshaped to B time-steps.

The training of the GESME-Net involves minimizing the mean squared error between the predicted values and the ground truth, which is achieved through the task-specific loss function f_{Loss}^{p} in Eq. (5.17). The objective of the overall loss function f_{Loss} applied in the architecture including regularization terms can be expressed as Eq. (4.5):

$$f_{Loss}^{p}\left(\boldsymbol{O}_t,\boldsymbol{A}_t\right)=\left\|\boldsymbol{O}_t-\boldsymbol{A}_t\right\|_2^2 \tag{5.17}$$

$$\min_{W^{(A)},W^{(FI)},b} f_{Loss}=\sum_{p=1}^{n} f_{Loss}^{p}+\alpha\left\|\boldsymbol{W}^{(A)}\right\|_2^2+\beta\left\|\boldsymbol{W}^{(FI)}\right\|_1 \tag{5.18}$$

where $\boldsymbol{A}_t\in\mathbb{R}^{N}$ is the ground truth vector for a task p, $\boldsymbol{W}^{(A)}$ refers to all parameters of the GESME-Net except the feature importance layer parameter $\boldsymbol{W}^{(FI)}$, and α,β are the regularization parameters. The L1- and L2-norm of regularization are utilized in accordance

with the requirements of the feature weighting layer, which also assists in avoiding overfitting issues.

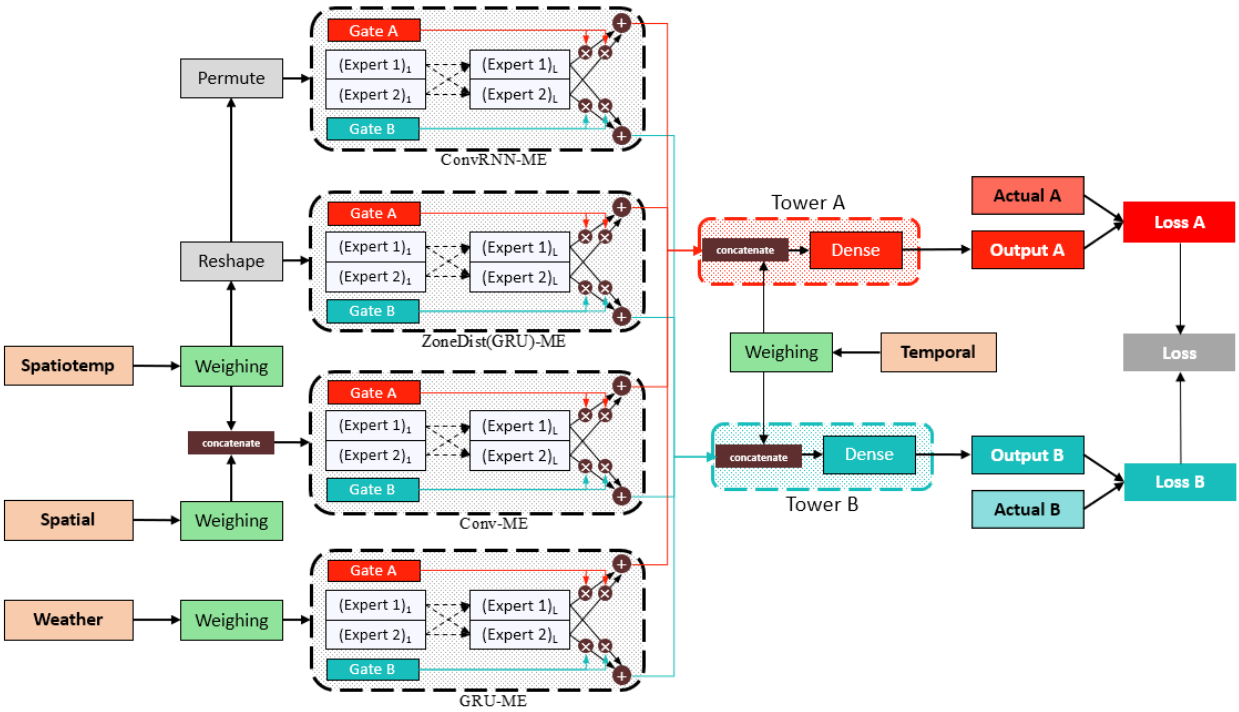


Figure 5.3. GESME-Net Architecture

5.3 Data Preparation, Experiments, and Model Evaluation

In this section, the proposed GESME-Net is tested with real-world data and the findings from the results are discussed. Furthermore, model ablation and model interpretation of the proposed GESME-Net is also provided. Finally, a sensitivity analysis of the GESME-Net hyperparameters is provided at the end of the section.

5.3.1 Data Description and Preprocessing

In this section, the data description and preprocessing for the two forecasting scenarios are provided. The first scenario involves jointly predicting demand across cities with data from

Chengdu and Xian and the second scenario involves jointly predicting demand and supply-demand gap with data from Beijing.

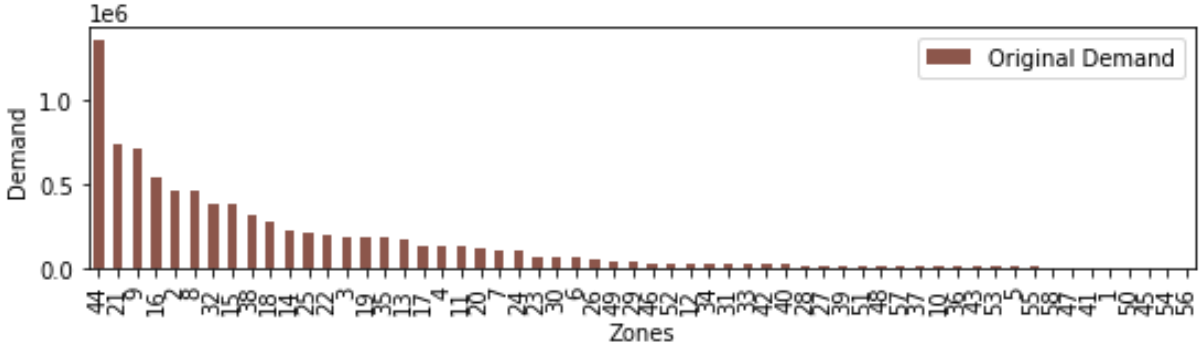
5.3.1.1 Scenario-1: Forecasting Different Tasks in a City

Publicly released ride-hailing dataset [116] of Didi Chuxing from Beijing is selected for simultaneously forecasting two spatio-temporal tasks, i.e., demand and supply-demand gap. Didi Chuxing divided each city into several square zones by applying geohashing and identified each zone with a unique ID, anonymizing the adjacency information among the zones.

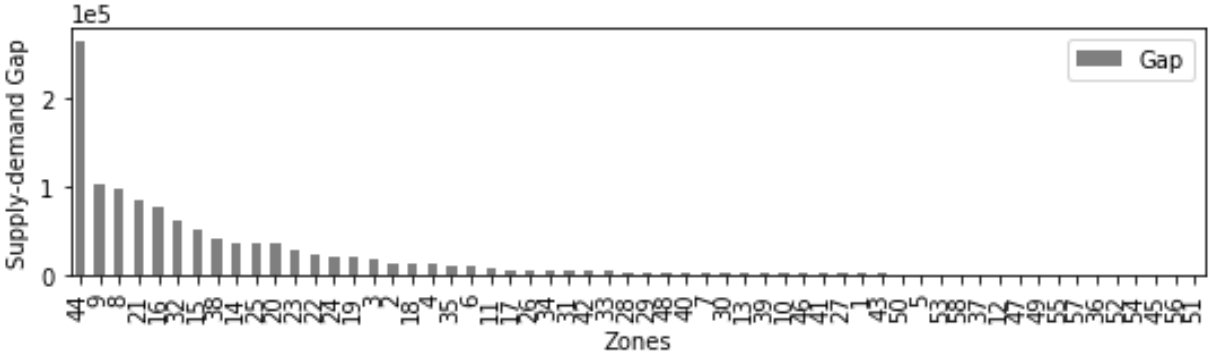
We utilized the Beijing dataset spanning from 1st January 2016 to 20th January 2016. To construct time-series data for each zone, the total timespan is divided into equal interval time-slots. Considering the small length of the datasets and focusing on short-term forecasting, each day in the datasets is therefore divided into 144 time-slots of 10 minutes interval. Furthermore, the total time-slots in a zone are split into training, validation, and testing sets for the experiments. Around 30% of the time-slots are reserved for validation and testing, and the rest of the data are used in training the models.

Both the datasets contain information around 8.5 million ride-hailing orders. The order information provides anonymized information of each order including driver ID, passenger ID, and, trip origin and destination geohashes. Besides, date and time of the orders are also available in the datasets. The unfulfilled orders are marked by a ‘null’ driver ID, which is useful information for calculating the supply-demand gaps in a zone for a time-slot. Furthermore, in order to find the demand and quantity supplied of each zone from the order information, the orders including ‘null’ driver IDs and orders excluding ‘null’ driver IDs are aggregated, respectively, for a timeslot. For both the datasets, around 50 percent of the time-slots are found to have zero supply-demand gaps, around 17 percent of which are due to zero

original demand, indicating that orders of around 33 percent time-slots are fully matched by the platform. The zone-wise distribution of the original demand and the supply-demand gap, is shown in Figure 5.4 (a) and Figure 5.4 (b) respectively. In general, relatively higher original demand zones tend to show a higher supply-demand gap in both datasets.



(a) Distribution of original demand across zones



(b) Distribution of supply-demand gap across zones

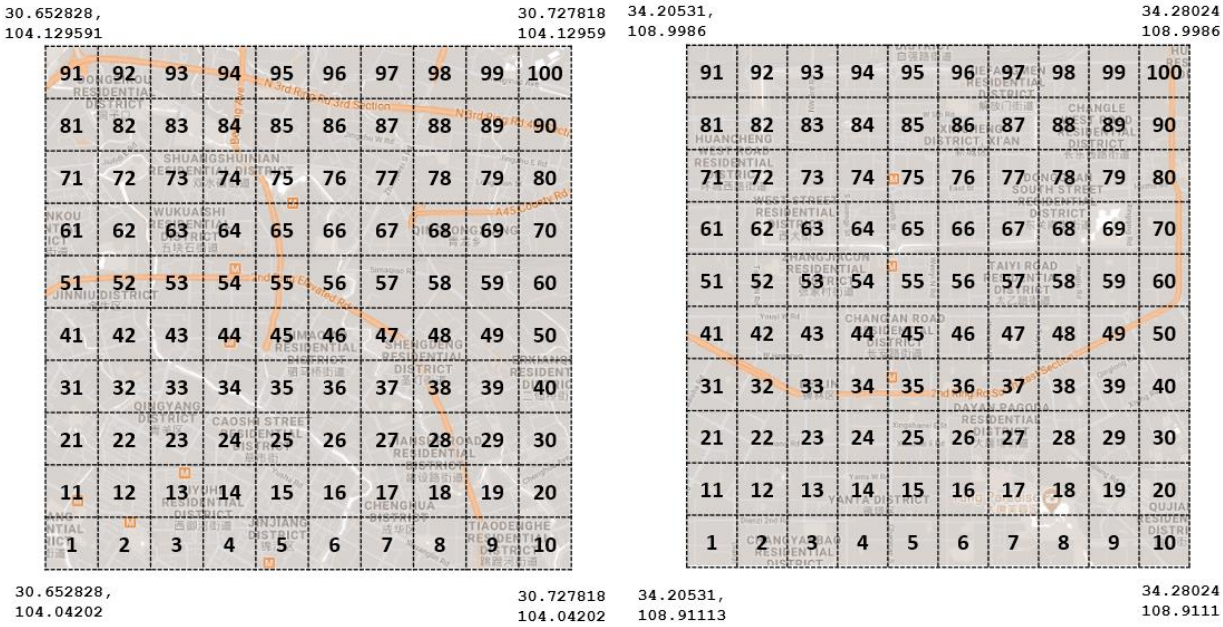
Figure 5.4. Distribution of demand and supply-demand gap in Beijing

5.3.1.2 Scenario-2: Forecasting Same Task Across Different Cities

Trajectory datasets from Chengdu and Xian under Didi Chuxing GAIA open dataset initiative [96] are selected for simultaneously learning same spatio-temporal task, i.e., demand across different cities. Since the datasets provided raw trajectories only, data processing is used to

extract the spatio-temporal variables and external weather and POI datasets are used to extract the required weather and POI variables.

We divided both Chengdu and Xian city into 10×10 square zones as shown in Figure 3. (a) and Figure 3. (b) respectively. Both datasets span from 1st October 2016 to 30th November 2016. Each day in the datasets is divided into 96 time-slots of 15 minutes interval. Furthermore, the total time-slots in a zone are split into training, validation, and testing sets for the experiments. Around 30% of the time-slots are reserved for validation and testing, and the rest of the data are used in training the models.



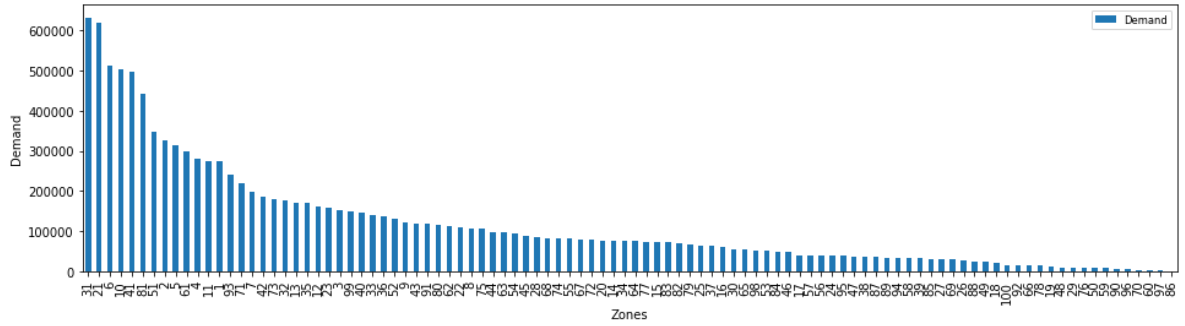
(a) Chengdu (b) Xian

Figure 5.5. Spatial partitioning of Chengdu and Xian

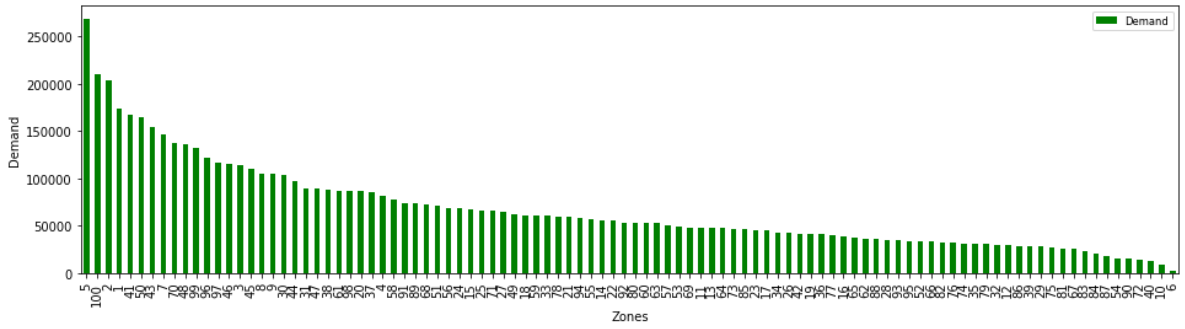
The Chengdu and Xian datasets contain anonymized trajectories of around 11.75 and 6.72 million ride-hailing trips respectively. The accessibility of a zone in a time-slot is calculated by counting the trajectories that fall in that zone. The average speed of a ride-hailing vehicle in a zone is found by extracting speed of each ride-hailing vehicle, calculating the distance from the trajectory portion that falls in that zone and dividing with the corresponding time to

cover that distance. The demand of a zone is calculated by extracting starting point of the trajectories and aggregating them in the respective time-slot. For both the datasets, around 12 percent of the time-slots are found to be zero-demand time-slots. The zone-wise distribution of the demand for Chengdu and Xian are shown in Figure 3. (a) and Figure 3. (b) respectively.

City-level weather variables (i.e., weather category, temperature, humidity, visibility, cloud cover, and wind speed) for Chengdu and Xian are collected from Dark Sky [97] at 15 minutes interval. The zone-wise time-invariant POI information for Chengdu and Xian are extracted from Gaode Map [98]. The zone-wise number of facilities for different POI types are aggregated to get the total POI per zone.



(a) Distribution of demand across zones of Chengdu



(b) Distribution of demand gap across zones of Xian

Figure 5.6. Distribution of demand in Chengdu and Xian

5.3.2 Model Evaluation

The performance of the proposed GESME-Net is compared to a set of benchmark algorithms. Furthermore, an ablation analysis of GESME-Net is conducted to justify increased model complexity. For a fair comparison, all models utilized the same lookback window, i.e., up to sixth previous time-slot for forecasting original demand and supply-demand gap in the Beijing dataset and up to ninth previous time-slot for forecasting demand in the Chengdu and Xian datasets. The settings of GESME-Net are decided by tuning of the hyperparameters. The finalized settings of the hyperparameters are presented in Table 5.1..

Table 5.1. Hyperparameter settings of GESME-Net

	Hyperparameter settings
(a) Feature Weighting Layer	Weight initialization: uniform; activation: linear
(b) Gated Convolutional Mixture of Experts	Layers: 2 layers; experts per layer: 2; filters: 25 in 1st layer; 50 in 2nd layer; filter length: 7-9 Weight initialization: uniform
(c) Gated Recurrent Mixture of Experts	Layers: 2 layers; experts per layer: 2; hidden units: 4 units per layer Weight initialization: uniform
(d) Gated Convolutional Recurrent Mixture of Experts	Layers: 2 layers; experts per layer: 2; filters: 50 in 1st layer; 100 in 2nd layer; filter length: 5 Weight initialization: uniform
(d) Fully Connected Layer	Weight initialization: uniform; activation: ReLU
(e) Model Training	Optimizer: Adam [99]; learning rate: 0.001; batch size: 32 Regularization: L1 = 0.001, L2 = 0.001; early stopping patience = 50-100 epochs

In order to assess the performance of GESME-Net, several machine learning models are considered such as GBM, XGBoost, RF, XRT, GLM, and ANN, which are extensively tuned by utilizing automated machine learning frameworks.

In addition to the abovementioned machine learning models, a number of deep learning models are considered as benchmarks. The following configurations are tested:

1) Spatio-temporal Mixture Network (SM-Net): The SM-Net is a special case of GESME-Net where a single network without subnetworks is utilized without any gating network, specialized to predict only one spatio-temporal task at a time.

2) Shared-bottom Spatio-temporal Mixture Network (SBSM-Net): The SBSM-Net is similar to SM-Net, however, contains a single network with shared parameters except the tower layer for multi-task learning.

3) Shared-gated Ensemble of Spatio-temporal Mixture of Experts Network (SESME-Net): The SESME-Net is a variation of GESME-Net, utilizing shared-gating instead of multi-gating for multi-task learning.

The performances of the models utilized in this study are evaluated with three metrics: mean absolute error (MAE), root mean squared error (RMSE), and symmetric mean absolute percentage error (sMAPE).

where $\boldsymbol{O}_i$ and $\boldsymbol{A}_i$ are the predicted vector and ground truth vector, respectively, at time-slot i in the test set with size n time-slots. Since the target value contains zero and sMAPE produces inaccurate statistics when encountered with zero, therefore, a modified sMAPE [5] is utilized.

The proposed architecture is trained on a server with 4 Core (hyper-threaded) Xeon processor (2.30 GHz), 25 GB RAM, and a Tesla P-100 GPU. The GESME-Net and its variations are written in Python 3 using Keras [106] with Tensorflow [107] backend. All the machine learning benchmarks are implemented in H2O AutoML [108].

The performances of the GESME-Net and the benchmark models are reported in Table 5.2 and Table 5.3. The GESME-Net performs marginally better than the deep learning benchmarks for each task in both scenarios. Such small improvement is not unexpected since all deep learning benchmarks are variations of the GESME-Net. However, the GESME-Net has around 10 % and 8 % lower RMSE than the best machine learning benchmark GBM for forecasting scenario-1 and scenario-2, respectively. Furthermore, the GESME-Net improves the MAE by at least 6 % than the machine learning benchmarks for both scenarios, with a maximum of at least 12 % in the demand forecasting for Xian. The sMAPE improvement of the GESME-Net than the best machine learning benchmark GBM is found to be around 2-6 %. It is noteworthy to mention that the total training times of the multi-task learning models are equally divided among the forecasting tasks to facilitate comparison with the single-task learning models.

Table 5.2. Performance of GESME-Net and benchmark models for forecasting original demand and supply-demand gap in Beijing (Scenario-1)

	Metrics (Original demand)				**Metrics (Supply-demand gap)**			
Model	**MAE**	**RMSE**	**sMAPE**	**Time (s)**	**MAE**	**RMSE**	**sMAPE**	**Time (s)**
GESME-Net	6.41	16.83	0.1927	1198	3.42	14.91	0.2341	1198
SESME-Net	6.43	16.95	0.1959	1185	3.51	15.11	0.2401	1185
SBSM-	6.58	17.20	0.1982	697	3.60	15.63	0.2577	697

Net								
SM-Net	6.68	17.00	0.1903	1094	3.52	15.49	0.2641	1936
XGBoost	6.88	19.07	0.2009	3.68	3.70	16.74	0.2952	11.89
GBM	7.00	18.91	0.2186	3.99	3.77	16.53	0.3053	3.43
XRT	7.04	19.43	0.1994	78.40	3.91	17.07	0.3045	31.03
RF	7.23	22.18	0.2006	79.45	3.90	16.60	0.3044	37.32
GLM	7.65	19.78	0.2537	0.44	4.61	16.80	0.3998	1.01
ANN	14.48	25.57	0.4250	33.21	5.34	19.09	0.4671	114.74

Table 5.3. Performance of GESME-Net and benchmark models for forecasting demand in Chengdu and Xian (Scenario-2)

	Metrics (Demand-Chengdu)				**Metrics (Demand-Xian)**			
Model	**MAE**	**RMSE**	**sMAPE**	**Time (s)**	**MAE**	**RMSE**	**sMAPE**	**Time (s)**
GESME-Net	3.72	5.90	0.1574	4108	2.78	4.04	0.1632	4108
SESME-Net	3.75	6.02	0.1574	2349	2.88	4.11	0.1698	2349
SBSM-Net	3.82	6.18	0.1553	1523	2.84	4.11	0.1702	1523
SM-Net	3.74	5.93	0.1549	3079	2.87	4.20	0.1679	3263
XGBoost	4.15	6.53	0.1831	27.68	3.34	4.58	0.2120	26.13
GBM	4.06	6.40	0.1805	20.80	3.18	4.44	0.2023	11.43
XRT	4.33	6.76	0.1889	133.89	3.45	4.79	0.2127	126.96
RF	4.51	6.92	0.2000	114.41	3.25	4.68	0.1942	113.02
GLM	4.34	7.04	0.1843	1.68	3.17	4.58	0.1922	1.76
ANN	4.32	6.83	0.2009	22.22	3.06	4.45	0.1808	18.53

Model ablation of GESME-Net is also conducted by removing the model components one at a time. The results of ablation analysis are presented in Table 5.4 and Table 5.5. For both scenarios, the removal of model components deteriorates the MAE and RMSE, which indicates that each of the model components is essential for achieving best performance of the GESME-Net. The highest deterioration for scenario-1 is seen due to removal of Conv-ME, about 7.5 % increase in the RMSE in the original demand forecasting task and around 10.5 % increase in RMSE in the supply-demand forecasting task. However, the highest deterioration for scenario-2 is found due to removal of GRU-ME, around 4 % increase in the RMSE is seen for both Chengdu and Xian.

Table 5.4. Ablation analysis of GESME-Net in scenario-1

	Metrics (Original demand)			**Metrics (Supply-demand gap)**			
Removed	**MAE**	**RMSE**	**sMAPE**	**MAE**	**RMSE**	**sMAPE**	**Time (s)**
Weighting	6.51	17.04	0.1805	3.50	15.33	0.2464	3095
ConvRNN-ME	6.49	17.05	0.1990	3.46	15.32	0.2507	2770
Conv-ME	6.89	18.09	0.2150	3.84	16.48	0.2549	5097
ZoneDist(GRU)-ME	6.69	16.96	0.2180	3.49	15.12	0.2516	1901
GRU-ME	6.62	16.97	0.1943	3.52	14.93	0.2543	1471

Table 5.5. Ablation analysis of GESME-Net in scenario-2

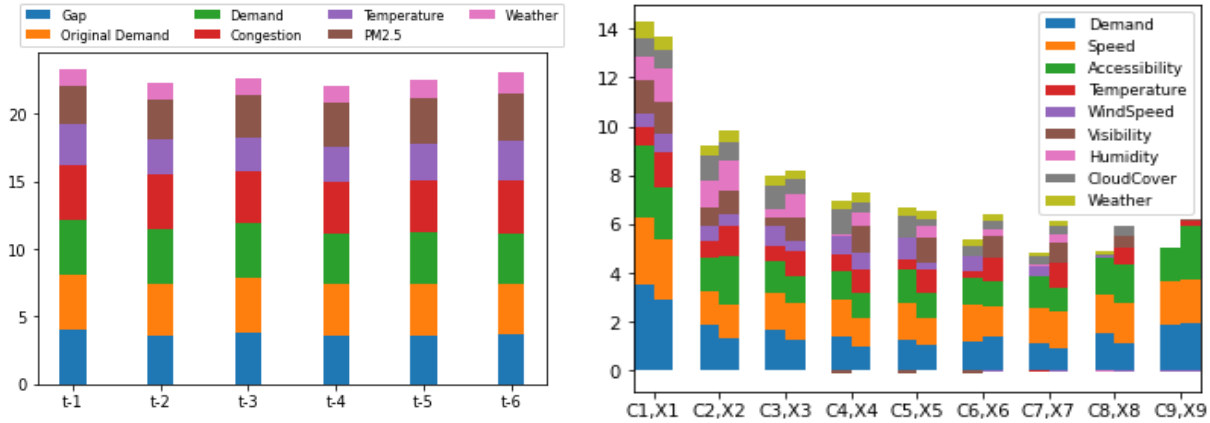
	Metrics (Demand-Chengdu)			**Metrics (Demand-Xian)**			
Removed	**MAE**	**RMSE**	**sMAPE**	**MAE**	**RMSE**	**sMAPE**	**Time (s)**
Weighting	3.78	6.02	0.1583	2.85	4.10	0.1746	3146
ConvRNN-ME	3.71	5.96	0.1520	2.86	4.12	0.1690	4293
Conv-ME	3.78	6.09	0.1560	2.91	4.18	0.1717	3990

ZoneDist(GRU)-ME	3.72	5.92	0.1548	2.85	4.13	0.1719	3686
GRU-ME	3.78	6.13	0.1553	2.97	4.21	0.1837	3457

5.3.3. Model Interpretations

In this study, the weights utilized by the feature weighting layers in the GESME-Net can be utilized to interpret the contribution of the input features in the forecasting model. In order to separately explain the contribution of the features temporally and spatially, the outputs of the feature weighting layers are averaged spatially and temporally, respectively.

Figure 5. (a) and Figure 5. (b) presents the spatially averaged feature weights across the time-slots for the spatio-temporal features and the weather features for scenario-1 and scenario-2 respectively. For scenario-1, the importance of the features is almost same in all the time-slots. However, within a time-slot, the spatio-temporal features have more effect than the weather features. For scenario-2, it is interesting to see that the historical values of the spatio-temporal variables, i.e., demand, speed and accessibility, show similar temporal importance patterns in both the cities, gradually decaying in the previous time-slots and then increasing again. However, this is not the case for the weather variables, importances of these variables gradually diminish in the previous time-slots and visibility even have negative association with prediction in the fourth, fifth and sixth previous time-slots from the prediction time-slot. This also indicates that weather variables have relatively less recurrent relationships with the prediction in multi-task learning across cities. Intuitively, weather variables can be city-specific and spatio-temporal variables can have commonalities across cities, which is well distinguished by the GESME-Net models.

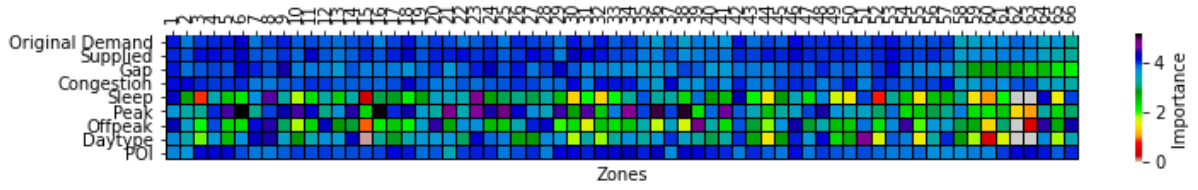


(a) Beijing

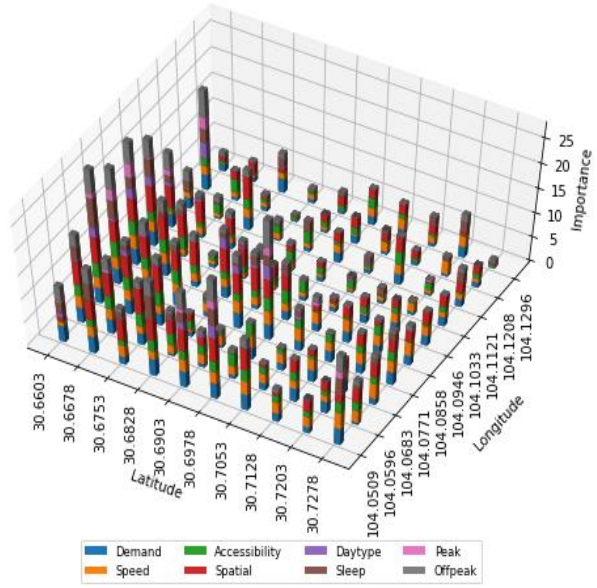
(b) C=Chengdu, X=Xian

Figure 5.7. Spatially averaged feature importance across previous time-slots

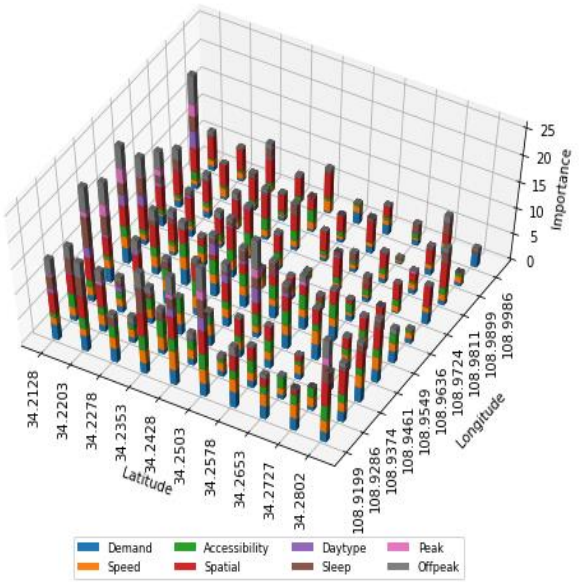
The temporally averaged weights of the spatio-temporal and the context features across the zones are presented in Figure 5. (a) for scenario-1 and in Figure 5.8 (b) and Figure 5.8 (c) for scenario-2. It is not unexpected that the importance of the spatio-temporal features and the context features are not uniformly weighted across the zones. For both forecasting scenarios, relatively more non-uniform effects are seen for the context features. For scenario-1, in general, it is seen that the features of the higher demand and higher supply-demand gap zones are more important to the GESME-Net for prediction than that of the lower demand and lower supply-demand gap zones. For scenario-2, the spatio-temporal and context variables of the south-east zones in both cities, usually found to be higher demand zones as evident from Figure 3., are the most important for prediction. The temporal context variables, i.e., peak, off-peak, and sleep, are relatively more emphasized in these zones by the GESME-Net models, which is interesting since it is generally expected that time-of-day characteristics are relatively more distinctive in the higher demand zones. Furthermore, GESME-Net also identified POI as an important spatial indicator for prediction in most of the zones and this is more evident in Xian than Chengdu.



(a) Beijing



(b) Chengdu

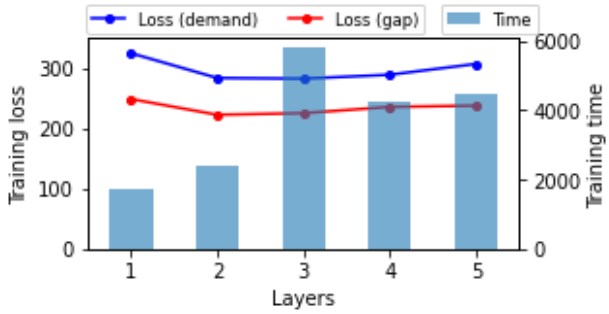


(c) Xian

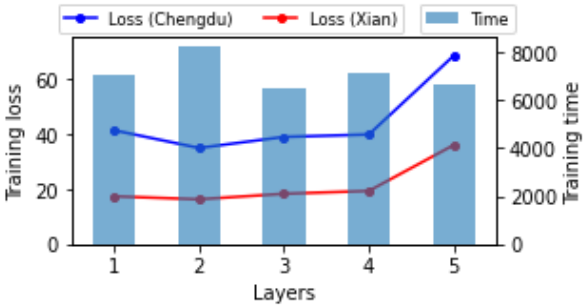
Figure 5.8. Temporally averaged feature importance across zones

5.3.4 Sensitivity Analysis

A sensitivity analysis of GESME-Net is conducted in terms of four hyperparameters: number of layers, number of filters in ConvRNN-ME and Conv-ME, filter size of ConvRNN-ME and Conv-ME, and the number of hidden units in GRU-ME and ZoneDist (GRU)-ME. In order to get the best performance for every hyperparameter configuration, all the experiments on hyperparameter tuning are conducted with early stopping.

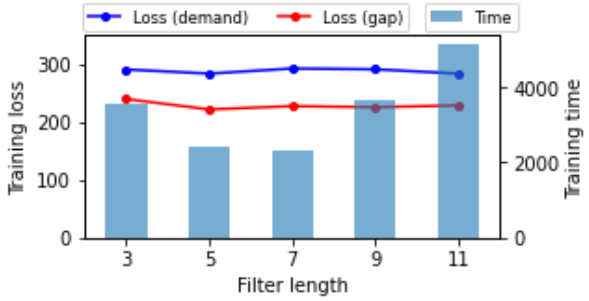


(a) Number of layers (Scenario-1)

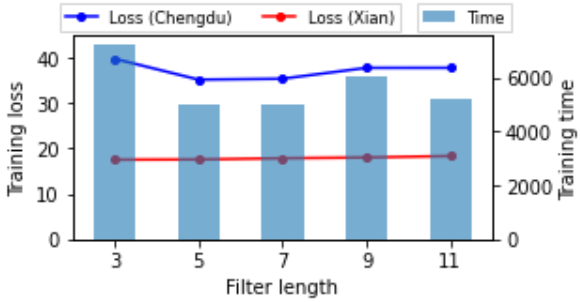


(b) Number of Layers (Scenario-2)

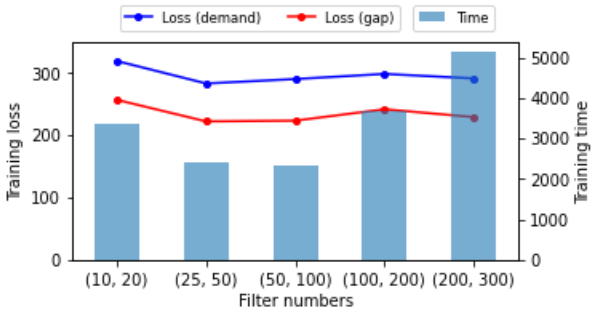
Figure 5.9. Sensitivity analysis for number of layers



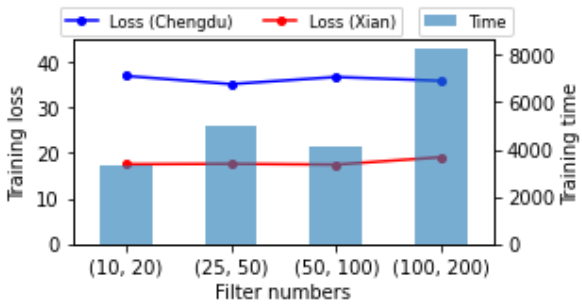
(a) Filter length (Scenario-1)



(b) Filter length (Scenario-2)



(c) Filter numbers (Scenario-1)

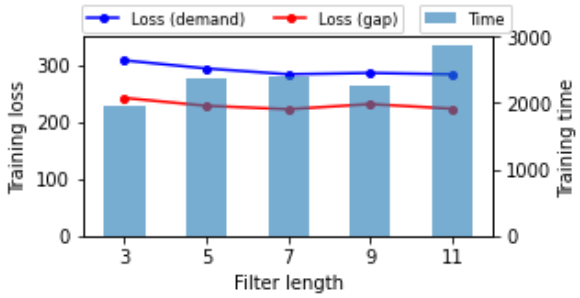


(d) Filter numbers (Scenario-2)

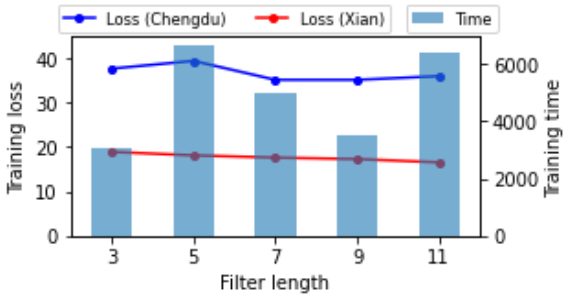
Figure 5.10. Sensitivity analysis for ConvRNN-ME

The variation of the training loss (i.e., mean squared error) of GESME-Net with respect to the number of layers for scenario-1 and scenario-2 are shown in Figure 3. (a) and Figure 3. (b) respectively. Losses for both scenarios reach the lowest for 2 layers and then rise again with

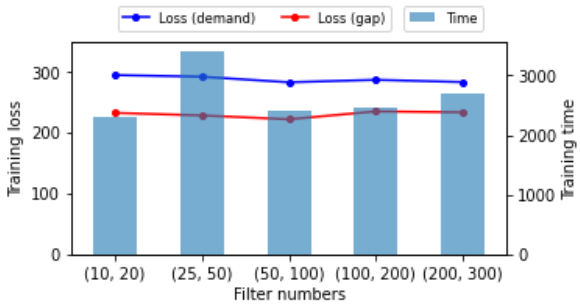
an increased number of layers. This result is not unintuitive since it is theoretically proven that neural networks with 2 hidden layers can approximate any continuous function [109]. Furthermore, the stability of GESME-Net for higher number of layers demonstrate its suitability for building deeper models.



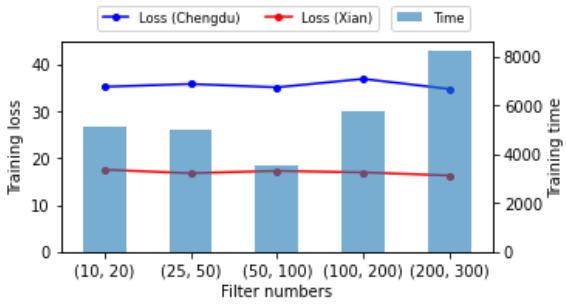
(a) Filter length (Scenario-1)



(b) Filter length (Scenario-2)



(c) Filter numbers (Scenario-1)

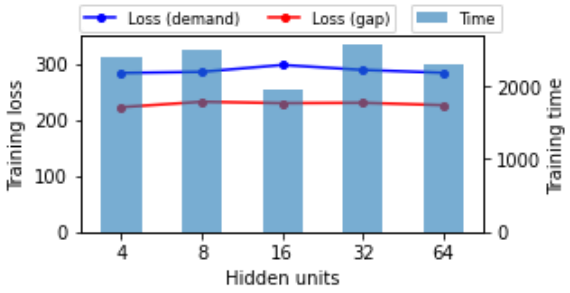


(d) Filter numbers (Scenario-2)

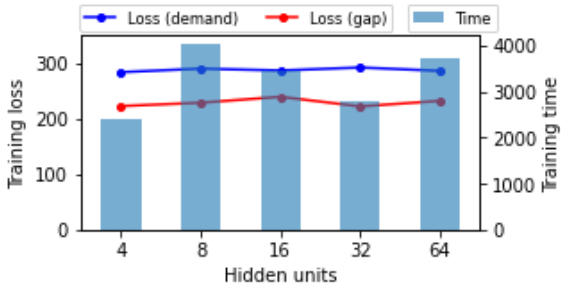
Figure 5.11. Sensitivity analysis for Conv-ME

The performance of the deep learning models with respect to the filter length and the number of filters in the first and second layer of the convolution utilized in ConvRNN-ME are shown in Figure 5.. It is evident that the combined loss for the tasks in both scenarios are lowest for filter length 5. Moreover, combined training losses of the proposed architecture for both scenarios gradually decrease and achieves the minimum for a combination of 25 filters in the first layer and 50 filters in the second layer. However, variable effects of filter numbers and filter length on the training losses are seen for convolution utilized in Conv-ME of scenario-1 and scenario-2, as shown in Figure 5.. The lowest combined loss is found for filter length 7

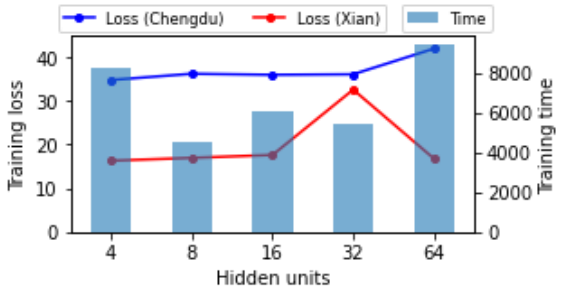
and filter combination (50,100) in scenario-1, whereas it is filter length 9 and filter combination (200,300) in scenario-2. It is evident that Conv-ME in the proposed architecture plays a major role in learning task relationships in spatio-temporal multi-task learning.



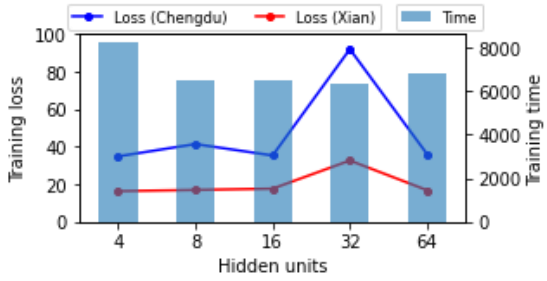
(a) GRU-ME (Scenario-1)



(b) GRU-ME (Scenario-2)



(c) ZoneDist(GRU)-ME (Scenario-1)



(d) ZoneDist(GRU)-ME (Scenario-2)

Figure 5.12. Sensitivity analysis for GRU-ME and ZoneDist(GRU)-ME

The sensitivity analysis for the variation of hidden units in GRU-ME and ZoneDist(GRU)-ME are shown in Figure 5.. For both scenarios, a common best performance achieved by 4 hidden units. Comparing with other hyperparameters, variations of the hidden units in GRU-ME show less fluctuations in training loss. However, abrupt fluctuations are seen for higher hidden units of ZoneDist(GRU)-ME due to overfitting.

CHAPTER 6: CONCLUSIONS AND RECOMMENDATIONS

This chapter gives an overview of the important findings and implications of this study. This is followed by limitations of the study as well as suggestions for future research.

6.1 Conclusions

In this study, three novel architectures are proposed for spatio-temporal forecasting in ride-hailing system. The proposed architectures bridge the gap between applying advanced deep learning methods and maintaining the privacy of the TNCs' data at the same time, provide a way for reducing the computational cost of spatio-temporal deep learning models for the TNCs, and demonstrates the viability of utilizing spatio-temporal multi-task learning architecture for jointly forecasting demand and supply-demand gap in a ride-hailing system. The conclusions from the developments made in this study are discussed below.

Firstly, the effectiveness of preserving spatial structure is investigated and a spatio-temporal deep learning baseline for spatio-temporal online taxi-hailing demand forecasting is developed in this study by combining 1DConvRNN, 1DCNN, and LSTM. The proposed baseline architecture is compared to deep learning architectures with increased model complexity through utilizing 2DCNN and 2DConvRNN/2DConvLSTM, where spatial structure is preserved by processing the zone-wise spatio-temporal and spatial data as image pixels. Furthermore, several machine learning algorithms such as GBM, XGBoost, RF, XRT, GLM and ANN are also considered as benchmarks for performance comparison along with the deep learning benchmarks. The models are validated on the real-world trajectory dataset from Chengdu provided by DiDi Chuxing, which is supplemented by external weather and POI datasets. The model based on the proposed architecture shows marginally better performance than the deep learning benchmarks. Furthermore, the proposed architecture

outperforms the machine learning benchmarks by a large margin, decreasing the RMSE at least 10 %. An ablation analysis of the model components and the feature categories are conducted, which shows that each of the model components and feature categories is essential for getting the best performance from the proposed architecture. Finally, sensitivity analysis of the deep learning models for varying hyperparameters is conducted to investigate the model performance in terms of model loss and training time, which clearly shows the superiority of the proposed architecture. The findings indicate that preserving spatial structure by processing zone-wise historical data as image pixels is not always necessary in spatio-temporal online taxi-hailing demand forecasting. A one-dimensional convolution applied to historical data with flattened zones can also better detect the patterns in spatio-temporal forecasting while being computationally cheaper. Furthermore, the proposed architecture can be utilized as a spatio-temporal baseline to develop more complex spatio-temporal deep learning models for spatio-temporal online taxi-hailing demand forecasting.

Secondly, a spatio-temporal deep learning architecture, FOCIR-Net, is proposed for forecasting both demand and supply-demand gap in a ride-hailing system with anonymized spatial adjacency information. The proposed architecture integrates feature importance layer with a spatio-temporal deep learning architecture composed of one-dimensional CNN and zone-distributed IndRNN. The architecture of FOCIR-Net processes different types of spatio-temporal, temporal and context features by spatially weighting them through the feature importance layer and learning spatial and temporal dependencies from the corresponding features through the one-dimensional CNN and zone-distributed IndRNN. The weights learned from the feature importance layer further assists in the ranking of the features and interpreting the model. The proposed FOCIR-Net is compared to several benchmark machine learning algorithms such as XGBoost, GBM, RF, XRT, GLM, and ANN. The models are tested with two real-world datasets of Didi from Beijing and Hangzhou, which shows the superiority of FOCIR-Net in terms of MAE, RMSE, and sMAPE in both demand and supply-demand forecasting tasks. An ablation analysis of the FOCIR-Net is conducted, which shows

that the processing of the spatio-temporal variables through the one-dimensional CNN is the most crucial part of the FOCIR-Net. Finally, interpretations of the models are provided based on the outputs of the feature importance layer. The proposed architecture demonstrates the applicability of spatio-temporal deep learning for forecasting with anonymized spatial adjacency information. Furthermore, model interpretation shows that the feature importance layer can assist the researchers to get a better understanding of the spatio-temporal deep learning models.

Finally, a spatio-temporal deep multi-task learning architecture, GESME-Net, is proposed for simultaneously forecasting multiple spatio-temporal tasks in a city as well as across cities. The proposed architecture integrates feature weighting layer with gated ensemble of spatio-temporal mixture of experts, i.e., Conv-ME, GRU-ME, and ConvRNN-ME, to model task relationships in multi-task learning as well as spatio-temporal dependencies. The weights learned from the feature importance layer assists in learning a common representation in multi-task learning, which can be further utilized in explaining the contribution of the features in the forecasting model. The proposed GESME-Net is compared against several benchmark models including task-specific spatio-temporal deep learning models, spatio-temporal multi-task learning models, and popular machine learning algorithms such as XGBoost, GBM, RF, XRT, GLM, and ANN. The models are tested in two multi-task learning scenarios with real world data from Didi Chuxing, which shows the superiority of GESME-Net in terms of MAE, RMSE, and sMAPE for simultaneously forecasting different spatio-temporal tasks in a city as well as same spatio-temporal task across different cities. An ablation analysis of the GESME-Net is conducted, which shows that each of the model components are crucial for getting best performance from the GESME-Net. Finally, interpretations of the spatio-temporal multi-task learning models are provided based on the outputs of the feature weighting layers. The proposed architecture demonstrates the development and application of spatio-temporal deep multi-task learning for simultaneously forecasting multiple spatio-temporal tasks in a city as well as across cities. Furthermore,

model interpretation from the feature weighting layer can assist in learning joint representation in spatio-temporal multi-task learning.

This study has several implications for the ride-hailing companies in Bangladesh. Although the architectures developed in this study are validated using data from China for forecasting demand and supply-demand gap, they can be adapted with data from the ride-hailing companies in Bangladesh by retraining the architectures. Furthermore, the transport planners of Bangladesh can also extend the proposed architectures for forecasting origin-destination matrices as a time-series forecasting problem. Most importantly, the multi-task learning architecture proposed in this study provides a way for reducing the computational burden in designing and maintaining spatio-temporal deep learning architectures through joint forecasting of spatio-temporal tasks that is economical for start-up ride-hailing companies in a developing country like Bangladesh.

6.2 Limitations and Future Research

This study has some limitations. Detailed information for some of the features such as weather categories, traffic congestion levels, and POI are unavailable in the Beijing and Hangzhou datasets due to confidentiality issues that limited us from utilizing more features. Furthermore, taxi trajectory data from a limited area of the Chengdu and Xian city provided, which restricted from large scale testing. Therefore, further investigation will be done when city-wide ride-hailing trajectory data becomes available.

There are several possible extensions from this study. The proposed spatio-temporal baseline architecture can be utilized with other spatio-temporal forecasting tasks such as traffic flow forecasting, traffic speed forecasting, and travel time forecasting. Furthermore, the proposed FOCIR-Net architecture can be tested against a large number of features in the future by extending the architecture for station-based platforms such as taxi and bike-sharing.

Importantly, the proposed GESME-Net architecture can be tested against a large number of spatio-temporal tasks in the future when data from several cities are available. Moreover, the GESME-Net components can be integrated with object detection frameworks for multi-class vehicle detection and natural language processing frameworks for traffic incident detection using multi-language data from twitter/facebook.

REFERENCES

[1] X. (Michael) Chen, M. Zahiri, and S. Zhang, "Understanding ridesplitting behavior of on-demand ride services: An ensemble learning approach," *Transp. Res. Part C Emerg. Technol.*, vol. 76, pp. 51–70, Mar. 2017.

[2] R. R. Clewlow and G. S. Mishra, "Disruptive Transportation: The Adoption, Utilization, and Impacts of Ride-Hailing in the United States," Oct. 2017.

[3] "DiDi Labs – Intelligent Transportation Technology and Security." [Online]. Available: http://www.didi-labs.com/.

[4] J. Ke, H. Zheng, H. Yang, and X. (Michael) Chen, "Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach," *Transp. Res. Part C Emerg. Technol.*, vol. 85, no. October, pp. 591–608, 2017.

[5] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, "Predicting taxi-passenger demand using streaming data," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1393–1402, Sep. 2013.

[6] M.-F. Chiang, T.-A. Hoang, and E.-P. Lim, "Where are the passengers?: a grid-based gaussian mixture model for taxi bookings," in *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '15*, 2015, pp. 1–10.

[7] D. Wang, W. Cao, J. Li, and J. Ye, "DeepSD: Supply-demand prediction for online car-hailing services using deep neural networks," in *Proceedings - International Conference on Data Engineering*, 2017, pp. 243–254.

[8] J. Ke *et al.*, "Hexagon-Based Convolutional Neural Network for Supply-Demand Forecasting of Ride-Sourcing Services," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 11, pp. 4160–4173, 2019.

[9] X. Geng *et al.*, "Spatiotemporal Multi-Graph Convolution Network for Ride-Hailing Demand Forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, vol. 33, pp. 3656–3663.

[10] M. G. McNally, "The Four-Step Model," in *Handbook of Transport Modelling*, Emerald Group Publishing Limited, 2007, pp. 35–53.

[11] A. Talvitie, "A direct demand model for downtown work trips," *Transportation (Amst).*, vol. 2, no. 2, pp. 121–152, Jul. 1973.

[12] L. B. Lave, “Demand for Intercity Passenger Transportation,” *Reg. Sci. J.*, vol. 12, no. 1, Apr. 1972.

[13] J. Choi, Y. J. Lee, T. Kim, and K. Sohn, “An analysis of Metro ridership at the station-to-station level in Seoul,” *Transportation (Amst).*, vol. 39, no. 3, pp. 705–722, May 2012.

[14] J. de D. Ortúzar and L. Willumsen, *Modelling transport*. John Wiley & Sons, 2011.

[15] M. G. Dagenais and M. J. I. Gaudry, “Can Aggregate Direct Travel Demand Models Work?,” Universite de Montreal, Departement de sciences economiques, 1986.

[16] X. Yan, X. Liu, and X. Zhao, “Using machine learning for direct demand modeling of ridesourcing services in Chicago,” *J. Transp. Geogr.*, vol. 83, p. 102661, Feb. 2020.

[17] C. Ding, X. Cao, and C. Liu, “How does the station-area built environment influence Metrorail ridership? Using gradient boosting decision trees to identify non-linear thresholds,” *J. Transp. Geogr.*, vol. 77, pp. 70–78, May 2019.

[18] H. Iseki, C. Liu, and G. Knaap, “The determinants of travel demand between rail stations: A direct transit demand model using multilevel analysis for the Washington D.C. Metrorail system,” *Transp. Res. Part A Policy Pract.*, vol. 116, pp. 635–649, Oct. 2018.

[19] L. Cheng, X. Chen, J. De Vos, X. Lai, and F. Witlox, “Applying a random forest method approach to model travel mode choice behavior,” *Travel Behav. Soc.*, vol. 14, pp. 1–10, Jan. 2019.

[20] X. Ma, J. Zhang, B. Du, C. Ding, and L. Sun, “Parallel Architecture of Convolutional Bi-Directional LSTM Neural Networks for Network-Wide Metro Ridership Prediction,” *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 6, pp. 2278–2288, Jun. 2019.

[21] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, “Short-term traffic forecasting: Where we are and where we’re going,” *Transp. Res. Part C Emerg. Technol.*, vol. 43, pp. 3–19, Jun. 2014.

[22] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[23] J. Baek and and K. Sohn, “Deep-Learning Architectures to Forecast Bus Ridership at the Stop and Stop-To-Stop Levels for Dense and Crowded Bus Networks,” *Appl. Artif. Intell.*, vol. 30, no. 9, pp. 861–885, Oct. 2016.

[24] J. Zhang, Y. Zheng, D. Qi, R. Li, and X. Yi, “DNN-based prediction model for spatio-temporal data,” in *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, 2016, pp. 1–4.

[25] J. Zhang, Y. Zheng, D. Qi, R. Li, X. Yi, and T. Li, “Predicting citywide crowd flows using deep spatio-temporal residual networks,” *Artif. Intell.*, vol. 259, pp. 147–166, Jun. 2018.

[26] X. Ouyang, C. Zhang, P. Zhou, H. Jiang, and S. Gong, “DeepSpace: An Online Deep Learning Framework for Mobile Big Data to Understand Human Mobility Patterns,” *arXiv Prepr. arXiv1610.07009*, Oct. 2016.

[27] D. Varshneya and G. Srinivasaraghavan, “Human trajectory prediction using spatially aware deep attention models,” *arXiv Prepr. arXiv1705.09436*, May 2017.

[28] Q. Chen, X. Song, H. Yamada, and R. Shibasaki, “Learning Deep Representation from Big and Heterogeneous Data for Traffic Accident Inference,” in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016, pp. 338–344.

[29] S. Alkheder, M. Taamneh, and S. Taamneh, “Severity Prediction of Traffic Accident Using an Artificial Neural Network,” *J. Forecast.*, vol. 36, no. 1, pp. 100–108, Jan. 2017.

[30] Y. Y. Chen, Y. Lv, Z. Li, and F. Y. Wang, “Long short-Term memory model for traffic congestion prediction with online open data,” in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2016, pp. 132–137.

[31] X. Ma, H. Yu, Y. Wang, and Y. Wang, “Large-Scale Transportation Network Congestion Evolution Prediction Using Deep Learning Theory,” *PLoS One*, vol. 10, no. 3, p. e0119044, Mar. 2015.

[32] W. Huang, G. Song, H. Hong, and K. Xie, “Deep architecture for traffic flow prediction: Deep belief networks with multitask learning,” *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2191–2201, Oct. 2014.

[33] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Y. Wang, “Traffic Flow Prediction with Big Data: A Deep Learning Approach,” *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, 2015.

[34] R. Fu, Z. Zhang, and L. Li, “Using LSTM and GRU neural network methods for traffic flow prediction,” in *Proceedings - 2016 31st Youth Academic Annual Conference of Chinese Association of Automation, YAC 2016*, 2017, pp. 324–328.

[35] Z. Zhao, W. Chen, X. Wu, P. C. V. Chen, and J. Liu, “LSTM network: A deep

learning approach for short-term traffic forecast," *IET Intell. Transp. Syst.*, vol. 11, no. 2, pp. 68–75, Mar. 2017.

[36] N. G. Polson and V. O. Sokolov, "Deep learning for short-term traffic flow prediction," *Transp. Res. Part C Emerg. Technol.*, vol. 79, pp. 1–17, Jun. 2017.

[37] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transp. Res. Part C Emerg. Technol.*, vol. 54, pp. 187–197, May 2015.

[38] Y. Jia, J. Wu, and Y. Du, "Traffic speed prediction using deep learning method," in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2016, pp. 1217–1222.

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

[40] C. Siripanpornchana, S. Panichpapiboon, and P. Chaovalit, "Travel-time prediction with deep learning," in *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, 2017, pp. 1859–1862.

[41] X. Gang *et al.*, "Continuous Travel Time Prediction for Transit Signal Priority Based on a Deep Network," in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2015, vol. 2015-Octob, pp. 523–528.

[42] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1681, Springer, Berlin, Heidelberg, 1999, pp. 319–345.

[43] R. J. Williams and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, Jun. 1989.

[44] I. Sutskever, G. Hinton, and G. Taylor, "The recurrent temporal restricted boltzmann machine," in *Advances in Neural Information Processing Systems 21 - Proceedings of the 2008 Conference*, 2009, pp. 1601–1608.

[45] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[46] H. Yao *et al.*, "Deep multi-view spatial-temporal network for taxi demand prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018, vol. 32, no. 1.

[47] X. Shi, Z. Chen, H. Wang, D. Y. Yeung, W. K. Wong, and W. C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems*, 2015, vol. 2015-Janua, pp. 802–810.

[48] H. Sak, A. Senior, and F. Beaufays, "Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition," *arXiv Prepr. arXiv1402.1128*, Feb. 2014.

[49] J. Collins, J. Sohl-Dickstein, and D. Sussillo, "Capacity and trainability in recurrent neural networks," in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017, pp. 1–17.

[50] W. Brendel and M. Bethge, "Approximating cnns with bag-of-local-features models works surprisingly well on imagenet," in *Seventh International Conference on Learning Representations*, 2019.

[51] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao, "Independently Recurrent Neural Network (IndRNN): Building A Longer and Deeper RNN," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5457–5466.

[52] R. Collobert and J. Weston, "A unified architecture for natural language processing," in *Proceedings of the 25th international conference on Machine learning - ICML '08*, 2008, pp. 160–167.

[53] M. Johnson *et al.*, "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation," *Trans. Assoc. Comput. Linguist.*, vol. 5, no. 1, pp. 339–351, 2017.

[54] M. L. Seltzer and J. Droppo, "Multi-task learning in deep neural networks for improved phoneme recognition," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2013, pp. 6965–6969.

[55] Z. Zhang, P. Luo, C. C. Loy, and X. Tang, "Facial landmark detection by deep multi-task learning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8694 LNCS, no. PART 6, Springer, Cham, 2014, pp. 94–108.

[56] Ł. Kaiser *et al.*, "One model to learn them all," *arXiv Prepr. arXiv1706.05137*, Jun. 2017.

[57] J. Ma, Z. Zhao, X. Yi, J. Chen, L. Hong, and E. H. Chi, "Modeling task relationships in multi-task learning with multi-gate mixture-of-experts," in *Proceedings of the ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018, pp. 1930–1939.

[58] X. Li *et al.*, "Prediction of urban human mobility using large-scale taxi traces and its applications," *Front. Comput. Sci. China*, vol. 6, no. 1, pp. 111–121, 2012.

[59] F. Wu, H. Wang, and Z. Li, "Interpreting traffic dynamics using ubiquitous urban data," in *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, 2016, pp. 1–4.

[60] Y. Tong *et al.*, "The Simpler the Better: A unified approach to predicting original taxi demands based on large-scale online platforms," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1653–1662.

[61] H. wen Chang, Y. chin Tai, and Y. jen J. Hsu, "Context-aware taxi demand hotspots prediction," *Int. J. Bus. Intell. Data Min.*, vol. 5, no. 1, pp. 3–18, 2010.

[62] K. Zhao, D. Khryashchev, J. Freire, C. Silva, and H. Vo, "Predicting taxi demand at high spatial resolution: Approaching the limit of predictability," in *Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016*, 2016, pp. 833–842.

[63] J. Liu, E. Cui, H. Hu, X. Chen, X. M. Chen, and F. Chen, "Short-term forecasting of emerging on-demand ride services," in *2017 4th International Conference on Transportation Information and Safety, ICTIS 2017 - Proceedings*, 2017, pp. 489–495.

[64] H. Wei, Y. Wang, T. Wo, Y. Liu, and J. Xu, "ZEST: A hybrid model on predicting passenger demand for chauffeured car service," in *International Conference on Information and Knowledge Management, Proceedings*, 2016, vol. 24-28-Octo, pp. 2203–2208.

[65] Y. Li, J. Lu, L. Zhang, and Y. Zhao, "Taxi booking mobile app order demand prediction based on short-term traffic forecasting," *Transp. Res. Rec.*, vol. 2634, no. 1, pp. 57–68, Jan. 2017.

[66] X. Qian, S. V Ukkusuri, C. Yang, and F. Yan, "Forecasting short-term taxi demand using boosting-GCRF," in *The 6th International Workshop on Urban Computing (UrbComp 2017)*, 2017, no. August.

[67] Y. Liu, Z. Liu, C. Lyu, and J. Ye, "Attention-Based Deep Ensemble Net for Large-Scale Online Taxi-Hailing Demand Prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4798–4807, Oct. 2020.

[68] C. Wang, P. Hao, G. Wu, X. Qi, and M. Barth, "Predicting the Number of Uber

Pickups by Deep Learning," in *Transportation Research Board 97th Annual Meeting*, 2018.

[69] J. Xu, R. Rahmatizadeh, L. Boloni, and D. Turgut, "Real-Time prediction of taxi demand using recurrent neural networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 8, pp. 2572–2581, Aug. 2018.

[70] W. Jiang and L. Zhang, "Geospatial data to images: A deep-learning framework for traffic forecasting," *Tsinghua Sci. Technol.*, vol. 24, no. 1, pp. 52–64, 2019.

[71] C. Wang, Y. Hou, and M. Barth, "Data-Driven Multi-step Demand Prediction for Ride-Hailing Services Using Convolutional Neural Network," in *Advances in Intelligent Systems and Computing*, vol. 944, no. April, Las Vegas, Nevada, 2020, pp. 11–22.

[72] X. Zhou, Y. Shen, Y. Zhu, and L. Huang, "Predicting multi-step citywide passenger demands using attention-based neural networks," in *WSDM 2018 - Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, 2018, vol. 2018-Febua, pp. 736–744.

[73] L. Bai, X. Wang, L. Yao, W. Liu, S. S. Kanhere, and Z. Yang, "Spatio-temporal graph convolutional and recurrent networks for citywide passenger demand prediction," in *International Conference on Information and Knowledge Management, Proceedings*, 2019, pp. 2293–2296.

[74] Y. Zhou, J. Li, H. Chen, Y. Wu, J. Wu, and L. Chen, "A spatiotemporal attention mechanism-based model for multi-step citywide passenger demand prediction," *Inf. Sci. (Ny).*, vol. 513, pp. 372–385, 2020.

[75] Y. Wang, T. Wo, H. Yin, J. Xu, H. Chen, and K. Zheng, "Origin-destination matrix prediction via graph convolution: A new perspective of passenger demand modeling," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019, pp. 1227–1235.

[76] Y. Xu and D. Li, "Incorporating graph attention and recurrent architectures for city-wide taxi demand prediction," *ISPRS Int. J. Geo-Information*, vol. 8, no. 9, 2019.

[77] G. Jin, Z. Xi, H. Sha, Y. Feng, and J. Huang, "Deep Multi-View Spatiotemporal Virtual Graph Neural Network for Significant Citywide Ride-hailing Demand Prediction," *arXiv Prepr. arXiv2007.15189*, 2020.

[78] W. Pian and Y. Wu, "Spatial-Temporal Dynamic Graph Attention Networks for Ride-hailing Demand Prediction," *arXiv Prepr. arXiv2006.05905*, 2020.

[79] G. Jin, Y. Cui, L. Zeng, H. Tang, Y. Feng, and J. Huang, “Urban ride-hailing demand prediction with multiple spatio-temporal information fusion network,” *Transp. Res. Part C Emerg. Technol.*, vol. 117, no. March, p. 102665, 2020.

[80] X. Zhang, X. Wang, W. Chen, J. Tao, W. Huang, and T. Wang, “A Taxi Gap Prediction Method via Double Ensemble Gradient Boosting Decision Tree,” in *Proceedings - 3rd IEEE International Conference on Big Data Security on Cloud, BigDataSecurity 2017, 3rd IEEE International Conference on High Performance and Smart Computing, HPSC 2017 and 2nd IEEE International Conference on Intelligent Data and Securit*, 2017, pp. 255–260.

[81] R. Wang, “Supply-demand Forecasting For a Ride-Hailing System,” University of California, Irvine, 2017.

[82] L. Ling, X. Lai, and L. Feng, “Forecasting the Gap Between Demand and Supply of E-hailing Vehicle in Large Scale of Network Based on Two-stage Model,” in *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019*, 2019, pp. 3880–3885.

[83] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-Decem, pp. 770–778.

[84] J. Li and Z. Wang, “Online car-hailing dispatch: Deep supply-demand gap forecast on spark,” in *2017 IEEE 2nd International Conference on Big Data Analysis, ICBDA 2017*, 2017, pp. 811–815.

[85] A. Ben Said and A. Erradi, “Multiview topological data analysis for crowdsourced service supply-demand gap prediction,” in *2020 International Wireless Communications and Mobile Computing, IWCMC 2020*, 2020, pp. 1818–1823.

[86] Z. Zhang, Y. Li, and H. Dong, “Multiple-Feature-Based Vehicle Supply-Demand Difference Prediction Method for Social Transportation,” *IEEE Trans. Comput. Soc. Syst.*, vol. 7, no. 4, pp. 1095–1103, Aug. 2020.

[87] L. Bai, L. Yao, S. S. Kanhere, Z. Yang, J. Chu, and X. Wang, “Passenger demand forecasting with multi-task convolutional recurrent neural networks,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11440 LNAI, Springer Verlag, 2019, pp. 29–42.

[88] K. Zhang, Z. Liu, and L. Zheng, “Short-Term Prediction of Passenger Demand in Multi-Zone Level: Temporal Convolutional Neural Network with Multi-Task Learning,” *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 4, pp. 1480–1490, 2020.

[89] J. Feng, M. Gluzman, and J. G. Dai, “Scalable deep reinforcement learning for ride-

hailing," *IEEE Control Syst. Lett.*, Sep. 2020.

[90] T. Kim, S. Sharda, X. Zhou, and R. M. Pendyala, "A stepwise interpretable machine learning framework using linear regression (LR) and long short-term memory (LSTM): City-wide demand-side prediction of yellow taxi and for-hire vehicle (FHV) service," *Transp. Res. Part C Emerg. Technol.*, vol. 120, no. September, p. 102786, 2020.

[91] L. Kuang, X. Yan, X. Tan, S. Li, and X. Yang, "Predicting taxi demand based on 3D convolutional neural network and multi-task learning," *Remote Sens.*, vol. 11, no. 11, p. 1265, 2019.

[92] H. Luo, J. Cai, K. Zhang, R. Xie, and L. Zheng, "A multi-task deep learning model for short-term taxi demand forecasting considering spatiotemporal dependences," *J. Traffic Transp. Eng. (English Ed.*, no. April, pp. 1–12, 2020.

[93] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," *Neural Comput.*, vol. 12, no. 10, pp. 2451–2471, 1999.

[94] B. Wang, Y. Lei, T. Yan, N. Li, and L. Guo, "Recurrent convolutional neural network: A new framework for remaining useful life prediction of machinery," *Neurocomputing*, vol. 379, pp. 117–129, Feb. 2020.

[95] Z. Yuan, X. Zhou, and T. Yang, "Hetero-ConvLSTM: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018, vol. 18, pp. 984–992.

[96] Didi, "GAIA Open Dataset," *DiDi Chuxing GAIA Open Dataset Initiative*, 2018. [Online]. Available: https://gaia.didichuxing.com.

[97] Apple, "Dark Sky API," *Dark Sky by Apple*, 2020. [Online]. Available: https://darksky.net/dev. [Accessed: 26-Sep-2020].

[98] S. I. Center, "Map POI (Point of Interest) data," *Peking University Open Research Data Platform*, 2017. [Online]. Available: https://doi.org/10.18170/DVN/WSXCNM. [Accessed: 26-Sep-2020].

[99] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[100] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *Ann. Stat.*, vol. 29, no. 5, pp. 1189–1232, 2001.

[101] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, vol. 13-17-Augu, pp. 785–794.

[102] A. Liaw and M. Wiener, “Classification and Regression by randomForest,” *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[103] P. Geurts, D. Ernst, and L. Wehenkel, “Extremely randomized trees,” *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, Apr. 2006.

[104] J. A. Nelder and R. W. M. Wedderburn, “Generalized Linear Models,” *J. R. Stat. Soc. Ser. A*, vol. 135, no. 3, pp. 370–384, May 1972.

[105] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[106] F. Chollet, “Keras: the Python deep learning API,” 2015. [Online]. Available: https://keras.io/. [Accessed: 04-Jul-2020].

[107] M. Abadi *et al.*, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems,” 2015. [Online]. Available: www.tensorflow.org. [Accessed: 04-Jul-2020].

[108] H2O.ai, “H2O AutoML.” 2017.

[109] J. T. Heaton, *Introduction to Neural Networks with Java*. Heaton Research, Inc., 2005.

[110] V. Borisov, J. Haug, and G. Kasneci, “CancelOut: A Layer for Feature Selection in Deep Neural Networks,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11728 LNCS, Springer Verlag, 2019, pp. 72–83.

[111] Y. Y. Lu, Y. Fan, J. Lv, and W. S. Noble, “Deeppink: Reproducible feature selection in deep neural networks,” in *Advances in Neural Information Processing Systems*, 2018, pp. 8676–8686.

[112] Y. Li, C. Y. Chen, and W. W. Wasserman, “Deep feature selection: Theory and application to identify enhancers and promoters,” *J. Comput. Biol.*, vol. 23, no. 5, pp. 322–336, 2016.

[113] B. Shen, M. Liu, X. Liang, W. Zheng, Y. Ouyang, and K. M. Carley, “StepDeep: A novel spatial-temporal mobility event prediction framework based on deep neural network,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018, pp. 724–733.

[114] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in Neural Information Processing Systems*, 2014, pp. 3104–3112.

[115] A. LeNail, “NN-SVG: Publication-Ready Neural Network Architecture Schematics,” *J. Open Source Softw.*, vol. 4, no. 33, p. 747, Jan. 2019.

[116] Didi, “Didi Di-Tech Challenge Algorithm Competition,” *Didi*, 2016. [Online]. Available: http://web.archive.org/web/20170311212917/http://research.xiaojukeji.com/competition/main.action?competitionId=DiTech2016.

[117] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, “Adaptive Mixtures of Local Experts,” *Neural Comput.*, vol. 3, no. 1, pp. 79–87, Feb. 1991.

[118] D. Eigen, M. A. Ranzato, and I. Sutskever, “Learning factored representations in a deep mixture of experts,” in *2nd International Conference on Learning Representations, ICLR 2014 - Workshop Track Proceedings*, 2014.

[119] R. Caruana, “Multitask Learning,” *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997.

[120] S. Ruder, “An Overview of Multi-Task Learning in Deep Neural Networks,” *arXiv Prepr. arXiv1706.05098*, Jun. 2017.

[121] K. Cho *et al.*, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” in *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2014, pp. 1724–1734.

[122] R. Jozefowicz, W. Zaremba, and I. Sutskever, “An empirical exploration of Recurrent Network architectures,” in *32nd International Conference on Machine Learning, ICML 2015*, 2015, vol. 3, pp. 2332–2340.

[123] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,” in *NIPS 2014 Workshop on Deep Learning*, 2014.

APPENDIX A: BEIJING AND HANGZHOU DATASET INFORMATION

The order data, weather data, traffic congestion data, and POI data are available in the raw datasets of Beijing and Hangzhou. All sensitive data has been anonymized.

A.1 Order Data

The order data shows the basic information of an order, including the passenger and the driver (if driver_id =NULL, it means the order was not answered by any driver), place of origin, destination, price and time. The fields order_id, driver_id, passenger_id, start_hash, and dest_hash are made not sensitive.

Table A.1. Order Information

Field	Type	Example	Meaning
order_id	string	70fc7c2bd2caf386bb 50f8fd5dfef0cf	Order ID (anonymized)
driver_id	string	56018323b921dd2c5 444f98fb45509de	Driver ID (anonymized)
passenger_id	string	238de35f44bbe8a67 bdea86a5b0f4719	User ID (anonymized)
start_district_ hash	string	d4ec2125aff74eded2 07d2d915ef682f	Departure (anonymized)
dest_district_ hash	string	929ec6c160e6f52c2 0a4217c7978f681	Destination (anonymized)
time	string	2016-01-15 00:35:11	Timestamp of the order

A.2 POI Data

The POI data shows the attributes of a district, such as the number of different facilities. For example, 2#1:22 means in this district, there are 22 facilities of the facility class 2#1. 2#1 means the first level class is 2 and the second level is 1, such as entertainment#theater, shopping#home appliance, sports#others.

Table A.2. POI Information

Field	Type	Example	Meaning
district_hash	string	74c1c25f4b283fa74a5514307b0d0278	District hash (anonymized)
poi_class	string	1#1:41 2#1:22 2#2:32	POI class and its number (anonymized)

A.3 Traffic Congestion Data

The traffic congestion data shows the overall traffic status on the road in a district, including the number of roads at different traffic congestion levels in different time periods and different districts. Higher values mean heavier traffic.

Table A.3. Traffic Congestion Information

Field	Type	Example	Meaning
district_hash	string	1ecbb52d73c522f184a6fc53128b1ea1	Hash value of the district (anonymized)

tj_level	string	1:231 2:33 3:13 4:10	Number of road sections at different congestion levels (anonymized)
tj_time	string	2016-01-15 00:35:11	Timestamp

A.4 Weather Data

The weather data shows the weather information every 10 minutes. The weather field gives the weather conditions such as sunny, rainy, and foggy etc; all sensitive information has been removed. The unit of temperature is Celsius degree, and PM2.5 is the level of air pollution.

Table A.4. Weather Information

Field	Type	Example	Meaning
time	string	2016-01-15 00:35:11	Timestamp
weather	int	7	Weather condition
temperature	double	-9	Temperature (°C)
PM2.5	double	66	pm2.5

APPENDIX B: CHENGDU AND XIAN DATASET INFORMATION

The route data, weather data, and POI data are available in the raw datasets of Chengdu and Xian. All sensitive data has been anonymized.

B.1 Route Data

The route data contains trajectory data of DiDi drivers. The measurement interval of the track points is approximately 2-4 seconds. The driver and trip order information are encrypted and anonymized.

Table B.1. Route Information

Field	Type	Example	Meaning
driver_id	string	glox.jrrlltBMvCh8nxqktdr2dtopmlH	Driver ID (anonymized)
order_id	string	jkkt8kxniovIFuns9qrrlvst@iqnpkwz	Order ID (anonymized)
timestamp	int	1501584540	Unix timestamp
longitude	double	104.04392	GCJ-02 Coordinate
latitude	double	30.66744	GCJ-02 Coordinate

B.2 POI Data

The POI data contains anonymized POI ID, type of the facility, longitude and latitude.

Table B.2. POI Information

Field	Type	Example	Meaning
poi_id	string	j#20180707#62b799bf5a15f132dfe	POI ID

		1b0536a65f723	(anonymized)
type	string	Road ancillary facilities	POI type
longitude	double	104.04392	GCJ-02 Coordinate
latitude	double	30.66744	GCJ-02 Coordinate

B.3 Weather Data

The weather data shows the weather information every 10 minutes. The weather information includes the weather conditions such as sunny, rainy, and foggy etc; temperature and dew point in degree Fahrenheit, visibility in kilometer, wind speed in kilometer per hour, cloud cover, and humidity.

Table B.3. Weather Information

Field	Type	Example	Meaning
timestamp	int	1501584540	Unix timestamp
weather	string	Foggy	Weather condition
temperature	double	66.1	Temperature (°F)
cloudCover	double	0.7	Cloud cover
dewPoint	double	60.77	Dew point (°F)
humidity	double	0.83	Humidity
visibility	double	9.84	Visibility (km)
windSpeed	double	5.59	Wind speed (kph)