Spatial-Temporal Generative Adversarial Learning

A Dissertation

Submitted to the Faculty

of

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Doctor of Philosophy

in

Data Science

by

Yingxue Zhang

April 18, 2022

APPROVED:

Professor Yanhua Li Worcester Polytechnic Institute Advisor	Professor Randy C. Paffenroth Worcester Polytechnic Institute Committee Member
Professor Xiangnan Kong Worcester Polytechnic Institute Committee Member	Professor Xun Zhou The University of Iowa External Committee Member
Professor Elke A. Rundensteiner Worcester Polytechnic Institute Head of Department	

Abstract

With the development of sensing and communication technologies, spatial-temporal big data has been widely generated and used in urban life, which helps to solve many problems related to smart cities, public safety, sustainability and business. However, it is challenging to deal with the spatial-temporal big data analytics problems (e.g., urban traffic estimation), because the data contains complex spatial-temporal dependencies, and is highly related to many other complicated factors. Given the large amount of spatial-temporal urban data, an important problem is how to successfully extract the complex spatial-temporal dependencies to solve diverse urban problems. In this dissertation, we will present an overview of my work that solves the spatial-temporal big data analytics problems in a deep generative adversarial perspective, and we will mainly introduce the spatial-temporal generative adversarial learning and its urban applications from the following 4 different perspectives.

1. Conditional Urban Traffic Estimation with Generative Adversarial Networks. The Conditional Urban Traffic Estimation problem aims to estimate the resulting traffic status prior to the urban construction plan. This problem is of great importance to urban development and transportation management, yet is very challenging due to the complex spatial-temporal dependencies and the relations between traffic and diverse urban conditions. To tackle this

problem, we propose three different generative adversarial networks including TrafficGAN, Curb-GAN and C^3-GAN.

2. Transferable Generative Adversarial Networks. Conventional methods for conditional traffic estimation usually focus on supervised settings, which require a large amount of labeled training data. However, in many urban planning applications, the large amount of traffic data in a new city can be hard or impossible to acquire. To tackle the conditional traffic estimation problem in data scarcity situations, we propose a novel STrans-GAN to extracts knowledge from multiple source cities to improve the estimation accuracy and transfer stability.

3. Learning Human Driving Strategies. This topic mainly focuses on the problem of human urban strategy analysis. This problem is hard to solve due to two major challenges: (1) data scarcity and (2) data heterogeneity. To solve the human urban strategy analysis problem in case of data scarcity and data heterogeneity, we design a novel learning paradigm — Spatial-Temporal Meta-GAIL (STM-GAIL), which can successfully learn diverse human urban strategies from heterogeneous human-generated spatial-temporal urban data.

4. Urban Traffic Dynamics Prediction. Predicting traffic dynamics is of great importance to urban development and transportation management. However, it is very challenging to solve this problem due to spatial-temporal dependencies and traffic uncertainties. In this topic, we solve the traffic dynamics prediction problem from Bayesian meta-learning perspective and propose a novel continuous spatial-temporal meta-learner (cST-ML).

Acknowledgements

I would like to give my warmest thanks to my adviser, Prof. Yanhua Li, for his endless help and guidance. I sincerely thank him for encouraging me to get through all the challenges in my research. I would like to thank my committee members, Prof. Xun Zhou, Prof. Xiangnan Kong and Prof. Randy C. Paffenroth for their generous encouragement and suggestions to my dissertation. I also thank all members in DSRG group for the opportunity to share my works, and they always provided bright ideas and critical comments.

I would like to thank my collaborators, Prof. Xun Zhou, Prof. Zhi-Li Zhang, Prof. Zhenming Liu, Dr. Jun Luo, Han Bao, for their novel research ideas, critical comments and productive collaborations. They provided a lot of exciting insights and incisive comments during our collaborations. I am also looking forward to future collaborations.

I thank all members of our research group, especially Guojun Wu, Menghai Pan, Huimin Ren, Xin Zhang, for helping me to improve my research projects during our discussions.

Last but not least, I would like to thank my family, Zeyu Zhu and Ping Xue for their love and supports all the way.

Publications

Publications Contributing to this Dissertation

In this section, I briefly summarize the publications pertinent to the topics covered in this dissertation.

Topic 1: Conditional Urban Traffic Estimation with Generative Adversarial Networks

Topic 1 of this dissertation deliberates on solving the conditional urban traffic estimation problem with novel generative adversarial networks. The publications in this topic are listed below.

1. [ICDM'19] **Yingxue Zhang**, Yanhua Li, Xun Zhou, Xiangnan Kong and Jun Luo. *TrafficGAN: Off-Deployment Traffic Estimation with Traffic Generative Adversarial Networks.* 2019 IEEE International Conference on Data Mining (ICDM).

2. [KDD'20] **Yingxue Zhang**, Yanhua Li, Xun Zhou, Xiangnan Kong and Jun Luo. *Curb-GAN: Conditional Urban Traffic Estimation through Spatio-Temporal Generative Adversarial Networks*. In Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD' 20).

3. [ICDM'21] **Yingxue Zhang**, Yanhua Li, Xun Zhou, Zhenming Liu and Jun Luo. *C^3-GAN: Complex-Condition-Controlled Urban Traffic Estimation through Generative Adversarial Networks.* 2021 IEEE International Conference on Data Mining (ICDM).

4. [TBD] **Yingxue Zhang**, Yanhua Li, Xun Zhou, Xiangnan Kong and Jun Luo. *Off-Deployment Traffic Estimation — A Traffic Generative Adversarial Networks Approach.* IEEE Transactions on Big Data (TBD).

5. [Under submission] **Yingxue Zhang**, Yanhua Li, Xun Zhou, Zhenming Liu and Jun Luo. *C^3-GAN$^+$: Complex-Condition-Controlled Generative Adversarial Networks with Enhanced Embeddings.* Under submission.

Topic 2: Transferable Generative Adversarial Networks

Topic 2 of this dissertation focuses on solving the data scarcity problem when estimating the urban traffic with transferable generative adversarial networks.

6. [Under submission] **Yingxue Zhang**, Yanhua Li, Xun Zhou, Xiangnan Kong and Jun Luo. *STrans-GAN: Spatially-Transferable Generative Adversarial Networks for Urban Traffic Estimation.* Under submission.

Topic 3: Learning Human Driving Strategies

Topic 3 of this dissertation focuses on learning diverse human driving strategies.

7. [Under submission] **Yingxue Zhang**, Yanhua Li, Xun Zhou, Xiangnan Kong and Jun Luo. *STM-GAIL: Spatial-Temporal Meta-GAIL for Learning Diverse Human Driving Strategies.* Under submission.

Topic 4: Urban Traffic Dynamics Prediction

Topic 4 studies the urban traffic dynamics prediction problem.

8. [ICDM'20]**Yingxue Zhang**, Yanhua Li, Xun Zhou and Jun Luo. *cST-ML: Continuous Spatial-Temporal Meta-Learning for Traffic Dynamics Prediction.* 2020 IEEE International Conference on Data Mining (ICDM).

9. [TIST]**Yingxue Zhang**, Yanhua Li, Xun Zhou, Jun Luo and Zhi-Li Zhang. *Urban Traffic Dynamics Prediction — A Continuous Spatial-Temporal Meta-Learning Approach*. ACM Transactions on Intelligent Systems and Technology (TIST).

Contents

Publications ii

List of Figures viii

List of Tables xii

1 Overview 1

1.1 Introduction 1

1.2 Dissertation Organization 5

2 Conditional Urban Traffic Estimation with Generative Adversarial Networks 7

2.1 Conditional Urban Traffic Estimation with TrafficGAN 7

2.1.1 Overview 7

2.1.2 Related Work 14

2.1.3 Methodology 15

2.1.4 Evaluation 25

2.2 Spatial-Temporal Generative Adversarial Networks 33

2.2.1 Overview 33

2.2.2 Related Work 38

2.2.3 Methodology 39

2.2.4 Evaluation 47

2.3 Complex-Condition-Controlled Urban Traffic Estimation 56
2.3.1 Overview 56
2.3.2 Related Work 60
2.3.3 Methodology 62
2.3.4 Evaluation 72

3 Transferable Generative Adversarial Networks 82

3.1 Spatially-Transferable Generative Adversarial Networks 82
3.1.1 Overview 82
3.1.2 Related Work 87
3.1.3 Methodology 89
3.1.4 Evaluation 98

4 Learning Human Driving Strategies 106

4.1 Spatial-Temporal Meta-GAIL 106
4.1.1 Overview 106
4.1.2 Related Work 112
4.1.3 Methodology 113
4.1.4 Evaluation 122

5 Urban Traffic Dynamics Prediction 130

5.1 Continuous Spatial-Temporal Meta-Learning 130
5.1.1 Overview 130
5.1.2 Related Work 137
5.1.3 Methodology 138
5.1.4 Evaluation 144

6 Conclusion and Future Work **155**

6.1 Conclusion 155

6.2 Future Work 158

6.2.1 Preference Reveal for Human Agents via Generative Adversarial Meta Learning. 158

6.2.2 Explainable Generative Adversarial Networks 159

6.2.3 Novel Application Domains in Urban Intelligence 159

References **160**

List of Figures

2.1 Traffic condition changes around Olympic Village in Beijing, China . . . 9

2.2 Travel demand and traffic distribution of region R 10

2.3 Off-deployment traffic estimation problem 11

2.4 Solution framework 13

2.5 Map gridding and regions illustration 16

2.6 Propagation rule of dynamic convolutional layer, f refers to traffic features 18

2.7 Convolutional filter comparison, $\lambda_1 > \lambda_2$ 20

2.8 TrafficGAN 21

2.9 TrafficGAN architecture 22

2.10 P_1 value changes with λ 26

2.11 P_1 comparisons over 8 target regions (seen and unseen) 28

2.12 Spatial patterns of 9 "unseen" regions 29

2.13 Traffic conditions of an "unseen" region 30

2.14 Traffic condition forecast. (a) is the target region covering the Longgang District; (b) is the actual traffic condition with current travel demand in the region; (c) is the forecast traffic condition with a higher expected travel demand, where more congestion appears; (d) indicates two possible reasons for (c); 32

2.15 Example of traffic estimation and evaluation for urban planning in Vaughan, Canada. 34
2.16 Insight of the framework. 35
2.17 Propagation rule of DyConv. 40
2.18 Filter comparisons of standard Conv and DyConv. 41
2.19 Example of 2 heads attention. 42
2.20 Overview of Curb-GAN. 43
2.21 Comparisons of selected models in consecutive time slots in traffic speed estimation. 50
2.22 Comparisons of selected models in consecutive time slots in taxi inflow estimation. 51
2.23 Traffic status visualizations. 52
2.24 Impact of parameters in traffic speed estimation. 54
2.25 Impact of parameters in taxi inflow estimation. 55
2.26 Traffic before & after building subway stations. 57
2.27 Solution framework. 59
2.28 C^3-GAN structure. 64
2.29 Visualizations of (1st row) traffic speed with bus route condition & (2nd row) taxi inflow with travel demand condition. 78
2.30 Loss plots of traffic speed and taxi inflow estimation regarding to bus route changes. 79
2.31 Impact of parameters on traffic speed and taxi inflow estimation with bus route conditions. 79

3.1 An example of conditional traffic estimation by transferring knowledge from multiple source cities (Shenzhen, Harbin, etc) to the target city (Xi'An). 83

3.2 Solution framework. 85
3.3 STrans-GAN Overview. 90
3.4 Visualizations of (1st row) traffic speed estimation in Xi'An & (2nd row) taxi inflow estimation in Chengdu. 100
3.5 Hyper-parameters studies on traffic speed estimation (Xi'An) and taxi inflow estimation (Chengdu). 102

4.1 Examples of taxi driver's decision-making process (left), data scarcity and heterogeneity challenges (right). 107
4.2 Solution framework. 108
4.3 Definition illustrations. 112
4.4 STM-GAIL structure. 115
4.5 Overall performance. 126
4.6 Performance on learning diverse strategies. 127
4.7 Impact of hyper-parameters on urban strategies learning with STM-GAIL. 127
4.8 Case studies: learned policies vs. ground-truth policies for two taxi drivers in two cases. 129

5.1 Illustration of traffic dynamics. 131
5.2 Problem illustration. 135
5.3 Insight of the framework. 136
5.4 Deterministic black-box meta-learning. 138
5.5 cST-ML performs as a rolling window. 141
5.6 Map gridding and target grid cells illustration. 145
5.7 Comparisons of models in 6 consecutive hours in traffic speed prediction. 150
5.8 Comparisons of models in 6 consecutive hours in taxi inflow prediction. . 151
5.9 Impact of parameters in traffic speed prediction. 151

5.10 Impact of parameters in taxi inflow prediction. 152

5.11 Traffic predictions of two target grid cells. 153

List of Tables

2.1 Statistics Comparisons for an “Unseen” Region 27
2.2 Statistics Comparisons for a “Seen” Region 27
2.3 Notations 37
2.4 Performance results on traffic speed estimation and taxi inflow estimation. 49
2.5 Variants of Curb-GAN evaluations. 49
2.6 Notations 58
2.7 Dataset descriptions. 74
2.8 Performance on task 1: traffic speed and taxi inflow estimation based on bus route changes. 77
2.9 Performance on task 2: traffic speed and taxi inflow estimation based on travel demand changes. 77

3.1 Dataset descriptions. 98
3.2 Performance on experiment 1: traffic speed and taxi inflow estimation in Xi’An. 98
3.3 Performance on experiment 2: traffic speed and taxi inflow estimation in Chengdu. 99
3.4 Ablation Study: traffic speed and taxi inflow estimation in Xi’An. 103

4.1 Notations 110

5.1 Notations. 134

5.2 Performance on traffic speed prediction and taxi inflow prediction. 148

1

Overview

1.1 Introduction

With the development of sensing and communication technologies, spatial-temporal big data has been widely generated and used in urban life, which helps to solve many problems related to smart cities, public safety, sustainability and business. However, it is challenging to deal with the spatial-temporal big data analytics problems (e.g., urban traffic estimation), because the data contains complex spatial-temporal dependencies, and is highly related to many other complicated factors. Given the large amount of spatial-temporal urban data, an important problem is how to successfully extract the complex spatial-temporal dependencies to solve diverse urban problems. In this dissertation, we will present an overview of my work that solves the spatial-temporal big data analytics problems in a deep generative adversarial perspective, and we will mainly introduce the spatial-temporal generative adversarial learning and its urban applications from the following 4 different perspectives.

1) Conditional Urban Traffic Estimation with Generative Adversarial Networks. The rapid progress of urbanization has expedited the process of urban planning, *e.g.*, new

residential, commercial areas, which in turn boosts the local travel demand. Given an urban development plan and the historical traffic observations over the road network, the Conditional Urban Traffic Estimation problem aims to estimate the resulting traffic status prior to the deployment of the plan. This problem is of great importance to urban development and transportation management, yet is very challenging because the plan would change the local travel demands drastically and the new travel demand pattern might be unprecedented in the historical data. To tackle this problem, we propose three different generative adversarial networks:

(i) TrafficGAN. The novel deep generative model TrafficGAN captures the shared patterns across spatial regions of how traffic conditions evolve according to travel demand changes and underlying road network structures. In particular, TrafficGAN captures the road network structures through a dynamic filter in the dynamic convolutional layer. We evaluate our TrafficGAN using a large-scale traffic data collected from Shenzhen, China. Results show that TrafficGAN can more accurately estimate the traffic conditions compared with all baselines. We also showcase that TrafficGAN can identify potential traffic issues in some regions and suggest possible reasons.

(ii) Curb-GAN. The proposed Curb-GAN can capture both spatial and temporal dependencies of urban traffic, it adopts and advances the conditional GAN structure through a few novel ideas: (1) dealing with various travel demands as the "conditions" and generating corresponding traffic estimations, (2) integrating dynamic convolutional layers to capture the local spatial auto-correlations along the underlying road networks, (3) employing self-attention mechanism to capture the temporal dependencies of the traffic across different time slots. Extensive experiments on two real-world spatio-temporal datasets demonstrate that our Curb-GAN outperforms major baseline methods in estimation accuracy under various conditions and can produce more meaningful estimations.

(iii) $C^3 - GAN$. Beside capturing spatial and temporal dependencies, we propose a

novel Complex-Condition-Controlled Generative Adversarial Network (C^3-GAN) for urban traffic estimation of a region under various complex conditions. C^3-GAN features the following three novel designs on top of standard cGAN model: (1) an embedding network mapping the complex conditions to a latent space to find high-quality representations of the urban conditions; (2) an inference network to enhance the relations between the embedded latent vectors and the traffic data; (3) a unique architecture to better capture the spatial dependencies of traffic, and a training algorithm for C^3-GAN to guarantee the network stability and performance. Extensive experiments on real-world datasets demonstrate that our C^3-GAN produces high-quality traffic estimations and outperforms state-of-the-art baseline methods.

2) Transferable Generative Adversarial Networks. Conditional traffic estimation is a vital problem in urban plan deployment, which can help evaluate urban construction plans and improve transportation efficiency. Conventional methods for conditional traffic estimation usually focus on supervised settings, which require a large amount of labeled training data. However, in many urban planning applications, the large amount of traffic data in a new city can be hard or impossible to acquire. To tackle the conditional traffic estimation problem in data scarcity situations, we formulate the problem as a spatial transfer generative learning problem. Compared to prior spatial transfer learning frameworks with only single source city, we propose to extracts knowledge from multiple source cities to improve the estimation accuracy and transfer stability, which is a technically more challenging task. As a solution, we propose a new cross-city conditional traffic estimation method — Spatially-Transferable Generative Adversarial Networks (STrans-GAN) with novel pre-training and fine-tuning algorithms. STrans-GAN preserves diverse traffic patterns from multiple source cities through traffic clustering, and incorporates meta-learning idea into the pre-training process to learn a well-generalized model. During fine-tuning, we propose to add a cluster matching regularizer to realize the flexible adaptation in differ-

ent scenarios. Through extensive experiments on multiple-city datasets, the effectiveness of STrans-GAN is proved.

3) Learning Human Driving Strategies. With large amounts of human-generated spatial-temporal urban data (*e.g.*, GPS trajectories of vehicles, passengers' trip data on buses and trains, *etc.*), human urban strategy analysis has become an important problem in many urban scenarios. This problem is hard to solve due to two major challenges: (1) data scarcity (*i.e.*, each human agent can only provide limited observations) and (2) data heterogeneity (*i.e.*, having mixed observations from many different human agents). Most of the existing works on this problem usually require a large amount of historical observations aiming to correctly infer a human agent's urban strategy and thus fail to properly address both challenges at the same time. To solve the human urban strategy analysis problem in case of data scarcity and data heterogeneity, we design a novel learning paradigm — Spatial-Temporal Meta-GAIL (STM-GAIL), which can successfully learn diverse human urban strategies from heterogeneous human-generated spatial-temporal urban data. STM-GAIL models the human decision processes as variable length Markov decision processes (VLMDPs) and incorporates the surrounding spatial feature patterns (*e.g.*, traffic volume patterns, *etc.*) into states to better capture the spatial-temporal dependencies of human decisions. Besides, STM-GAIL learns diverse human urban strategies from the meta-learning perspective, and can distinguish various human urban strategies by adding an inference network on top of the standard GAIL. STM-GAIL can be quickly adapted to a new human expert's urban strategy with a single trajectory. Extensive experiments on real-world human-generated spatial-temporal dataset are performed. STM-GAIL presents significant improvement over state-of-the-art baseline models when learning human urban strategies.

4) Urban Traffic Dynamics Prediction. Urban traffic status (*e.g.*, traffic speed and volume) is highly dynamic in nature, namely, varying across space and evolving over

time. Thus, predicting such traffic dynamics is of great importance to urban development and transportation management. However, it is very challenging to solve this problem due to spatial-temporal dependencies and traffic uncertainties. In this paper, we solve the traffic dynamics prediction problem from Bayesian meta-learning perspective and propose a novel continuous spatial-temporal meta-learner (cST-ML), which is trained on a distribution of traffic prediction tasks segmented by historical traffic data with the goal of learning a strategy that can be quickly adapted to related but unseen traffic prediction tasks. cST-ML tackles the traffic dynamics prediction challenges by advancing the Bayesian black-box meta-learning framework through the following new points: 1) cST-ML captures the dynamics of traffic prediction tasks using variational inference, and to better capture the temporal uncertainties within tasks, cST-ML performs as a rolling window within each task; 2) cST-ML has novel designs in architecture, where CNN and LSTM are embedded to capture the spatial-temporal dependencies between traffic status and traffic related features; 3) novel training and testing algorithms for cST-ML are designed. We also conduct experiments on two real-world traffic datasets (taxi inflow and traffic speed) to evaluate our proposed cST-ML. The experimental results verify that cST-ML can significantly improve the urban traffic prediction performance and outperform all baseline models especially when obvious traffic dynamics and temporal uncertainties are presented.

1.2 Dissertation Organization

The remainder of this dissertation is organized as follows. In **Chapter 2**, we introduce our work about Conditional Urban Traffic Estimation with Generative Adversarial Networks, which contains three different novel generative adversarial networks [1, 2, 3] for conditional urban traffic estimation. And in **Chapter 3**, our proposed transferable gener-

ative adversarial network can successfully solve the data scarcity problem in urban traffic estimation. In **Chapter 4** which is for our Learning Human Driving Strategies, a Spatial-temporal Meta-GAIL is designed to learn diverse human driving strategies. In **Chapter 5**, a novel continuous spatial-temporal meta-learning framework is designed for urban traffic dynamics prediction. A conclusion of this dissertation is drawn in **Chapter 6**.

For consistency and to allow the reader to easily jump from one topic to another, we present the detailed research for each topic in a largely self-contained set of sections following this particular pattern:

- Overview: Includes the introduction and preliminaries of the project (including definitions and notations);
- Related work: Reviews the related state-of-the-arts and analyzes their pros and cons, and also identifies the novelty of our work.
- Methodology: Introduces the details of the proposed method, including the objective function, detailed architecture and training and testing algorithms.
- Evaluation: Presents extensive experiments, this part contains the data description, introduction of baseline models, the evaluation metrics, the experimental settings and the final evaluation results.

2

Conditional Urban Traffic Estimation with Generative Adversarial Networks

2.1 Conditional Urban Traffic Estimation with Traffic-GAN

2.1.1 Overview

2.1.1.1 Introduction

Over the past a few decades, we have witnessed drastic urbanization at the global scale. It is reported that the world's urban population ratio has reached 54% in 2014, and it is projected that by 2050, two-thirds of the world population will live in urban areas[4].

With the rapid progress of urbanization, urban planning is becoming a vital problem concerning with resources allocation, urban transportation efficiency and living environment. The fast development of new residential and commercial areas always comes with population growth, which in turn increases the travel demands and the risk of worsening traffic conditions due to the overload of the transportation infrastructures. For example,

the Olympic Village was built in the northern area of Beijing for the 2008 Olympic Games with many new residential and commercial areas constructed in its nearby areas as illustrated in Fig. 5.1. The population in that region increased drastically after 2008, which significantly worsened the local traffic conditions [5]. This could have been avoided if more thorough and accurate traffic evaluation had been done before the constructions. Therefore, it is crucial to foresee both positive and negative impacts on traffic conditions before an urban construction plan is deployed. In our work, we refer to such a problem as "*off-deployment traffic estimation*" problem. Solving this problem is technically challenging, since no new data can be collected before deployment in an area, while old data collected before deployment fails to capture the traffic pattern changes.

The traffic estimation problem has been extensively studied in the literature [6, 7, 8, 9, 10]. These works use the historical traffic data of regions to build machine learning models that capture the correlations among the past traffic, environmental features and the future traffic. However, when predicting the traffic impact of a newly developed construction plan, these models will fail because they cannot capture the future traffic pattern changes caused by the new deployment plan due to the lack of training samples. Traditionally in civil engineering, agent-based simulation models [11] or physical models [12] are used to estimate the projected traffic volume after constructions. However, these models rely heavily on model choice and parameter settings, which are not transferable across urban regions.

In this paper, we propose a novel traffic generative adversarial network (TrafficGAN) to tackle the off-deployment traffic estimation problem. The proposed TrafficGAN captures the traffic correlations along the underlying road networks, and can estimate traffic conditions prior to deployment of a construction plan. Our **main contributions** are summarized as follows:

• We model the off-deployment traffic estimation problem as a traffic data generation

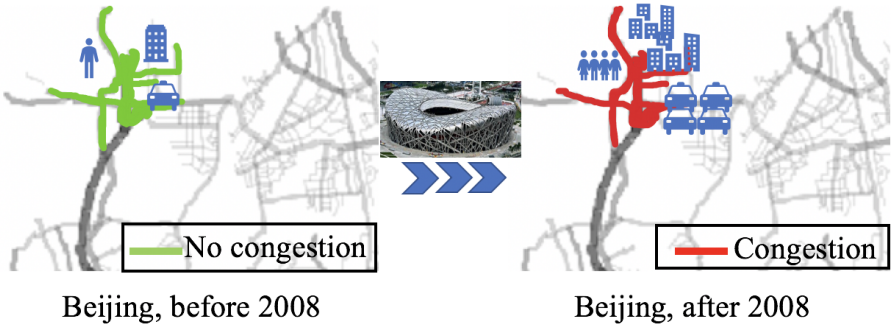


Figure 2.1: Traffic condition changes around Olympic Village in Beijing, China

problem, and propose a novel deep generative model – TrafficGAN, which captures the shared patterns across spatial regions of how traffic conditions evolve according to travel demand changes and underlying road network structures.

• We evaluate TrafficGAN using a large scale traffic data collected during 7/2016-12/2016 from Shenzhen, China. The unique dataset represents a wide range of regions with diverse travel demands and traffic conditions in both rural and urban areas. Our results demonstrate that our proposed TrafficGAN can accurately estimate the traffic conditions compared with all baselines. When applying TrafficGAN to a number of representative regions with higher (than their current) travel demands, we showcase the issues of those regions (in case of new construction plan deployed) and identify potential reasons of the issues.

2.1.1.2 Definitions

Urban planning, especially, governmental zoning[1], is a process of planning land use and development in a target region, in which certain land uses (*e.g.*, residential, commercial) are permitted or prohibited [13]. In this work, we focus on urban deployment and zoning plans when developing certain new residential or commercial areas in a target region, which will potentially promote the population size and influence the travel demand in

[1] `https://en.wikipedia.org/wiki/Zoning`

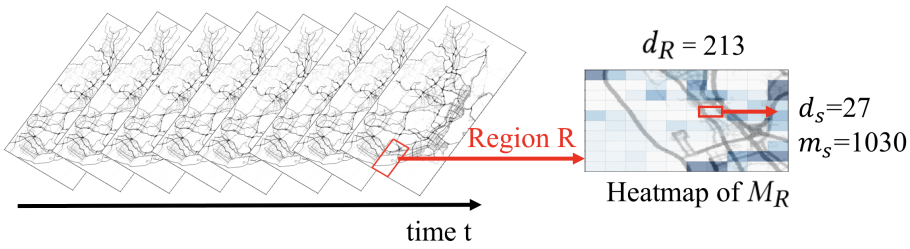


Figure 2.2: Travel demand and traffic distribution of region R

the region. Denote a city under planning as R_0, *e.g.*, Shenzhen City in China bounded by $[22.534°, 22.87°]$ in latitude and $[113.77°, 114.40°]$ in longitude. As defined below, we partition R_0 into *grid cells* (as the smallest granular region to characterize the traffic status) and form *target regions* R's as collections of grid cells.

Definition 1 (Grid cell s). The planning city R_0 is partitioned into N_0 grid cells with equal side-length in latitude and longitude, denoted as $\mathbb{S} = \{s_i\}$, where $1 \leq i \leq N_0, i \in \mathbb{N}$.

Definition 2 (Target region R). A target region R is a square geographic region in R_0, formed by $\ell \times \ell$ grid cells. Formally, $R = \langle s, \ell \rangle$ is uniquely defined by an anchor grid cell s on its top-left corner and a number ℓ of grid cells on the side[1].

In our study, grid cells are the minimum units where traffic status and travel demands are measured. Alternatively, urban planning and traffic estimation will be performed at a target region. As a result, a region is analogous to an "image", where grid cells are viewed as "pixels". Fig. 2.5a visualizes grid cells of Shenzhen, China and Fig. 2.5b illustrates the target region examples with $\ell = 10$.

Definition 3 (Travel demand of a grid cell and a target region). The travel demand of a geographic area captures the total number of departures in a period of time, *e.g.*, one hour interval. Thus, we denote the travel demand of a grid cell s as $d_s \in \mathbb{N}$. Given a target region R, $\boldsymbol{D}_R$ is an $\ell \times \ell$ matrix representing the travel demand distribution of

[1]Note that target regions can also be defined as rectangles rather than squares. For simplicity, we use square shape of target regions in this work.

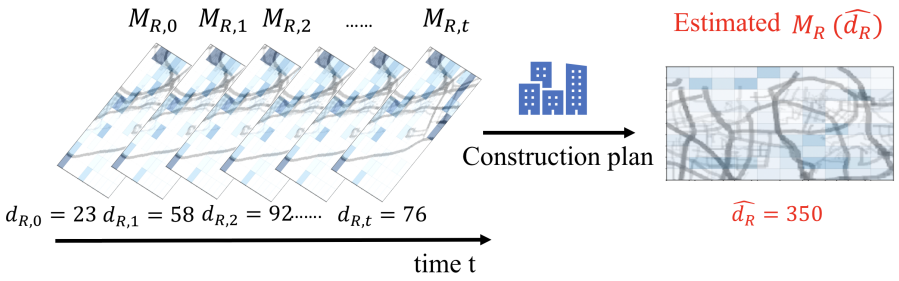


Figure 2.3: Off-deployment traffic estimation problem

all grid cells in R. Moreover, we denote the *total travel demand* of a target region R as $d_R \in \mathbb{N}$, which is the sum of travel demands in all grid cells within R, *i.e.*, $d_R = \sum_{s \in R} d_s$ = $\sum_{1 \leq i,j \leq l} \boldsymbol{D}_R(i,j)$.

In general, it is hard to obtain the total travel demand in a region including all transport modes. In this work, we use the demand for taxis to represent the regional travel demand, where many studies have shown that taxi demands represent the total demands quite well [14, 15].

Definition 4 (Traffic status of a grid cell and traffic distribution of a target region). Traffic status includes various measures representing the quality of traffic in a geographic region, such as average driving speed, traffic inflow/outflow, traffic volume, *etc*. Taking traffic inflow as an example, we denote m_s as the traffic inflow of grid cell s in a period of time. Similar, given a target region R with $\ell \times \ell$ grid cells, we denote an $\ell \times \ell$ matrix $\boldsymbol{M}_R$ as the traffic distribution in R.

Each element of $\boldsymbol{M}_R$ represents the taxi inflow in a grid cell. As shown in Fig. 2.2, the whole matrix $\boldsymbol{M}_R$ can be viewed and visualized as an "image" characterizing the underlying traffic distribution of a target region R, where each pixel represents a grid (*e.g.*, gird s, the small red box in the map).

Definition 5 (Urban deployment plan). An urban deployment plan in a target region R is referred to a plan to construct new residential or commercial areas in the region R without changing the road structures. As a part of the plan, the expected travel demand

after deployment is specified[1], denoted by $\hat{d}_R$.

Clearly, an urban deployment plan will lead to an updated regional travel demand $\hat{d}_R$, which in turn would significantly affect the regional traffic distribution $\boldsymbol{M}_R$. The goal of our work is to estimate $\boldsymbol{M}_R(\hat{d}_R)$, which reflects the potential traffic burden to be introduced by a deployment plan and can be used by the planning authorities to evaluate the pros and cons of urban deployment plans. The problem is formally defined as below.

Problem definition. Given a city area R_0 partitioned into grid cells $\mathbb{S}$, the citywide historical travel demands and traffic distributions $\boldsymbol{D}_{R_0,t}$ and $\boldsymbol{M}_{R_0,t}$ are available over a time span $1 \leq t \leq T$. For a target region $R = \langle s, \ell \rangle$ and a deployment plan in R with the expected travel demand $\hat{d}_R$, we aim to estimate the traffic distribution $\boldsymbol{M}_R(\hat{d}_R)$.

Fig. 5.2 illustrates an example of the problem. The series of matrices on the left are the historical traffic distributions and travel demands for each time slot. The map on the right is the estimated traffic distribution ($\boldsymbol{M}_R(\hat{d}_R)$) based on an expected travel demand $\hat{d}_R$=350 of the proposed plan.

2.1.1.3 Data Description

In effect, all kinds of personal vehicle data can be used, especially the GPS trajectories and other spatial-temporal data records. In this paper, we use two real-world traffic datasets, (1) taxi GPS data; (2) road map data. For consistency, all datasets are collected from the same time interval, *i.e.*, from Jul 1st to Dec 31st, 2016 in Shenzhen, China.

Taxi GPS data contains GPS records collected from taxis in Shenzhen, China from Jul 1st to Dec 31st, 2016. There are 17,877 taxis equipped with GPS sensors, each GPS sensor generates a GPS record every 40 seconds on average. Overall, a total number of 51,485,760 GPS records are collected each day, each record contains five key data fields including taxi ID, time stamp, passenger indicator, latitude and longitude. The passenger

[1]The expected travel demand $\hat{d}_R$ after deploying a construction plan is assumed given in this paper, which can be done by commonly used Four-Steps demand forecasting approaches in Civil Engineering [16].

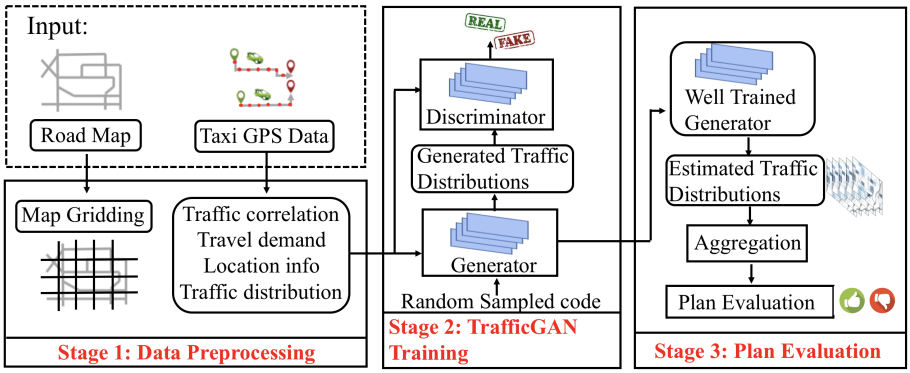


Figure 2.4: Solution framework

indicator field is a binary value indicating whether a passenger is on board.

Road map. In our study, we use the Google GeoCoding[1] to retrieve the bounding box of Shenzhen. The bounding box is defined between $22.534°$ to $22.87°$ in latitude and $113.77°$ to $114.40°$ in longitude. Shenzhen road map[2] is shown in Fig. 2.5.

2.1.1.4 Solution Framework

Fig. 4.2 outlines our off-deployment traffic estimation framework, which takes taxi GPS data and road map data as inputs, processes the data in three stages to get the output:

• **Stage 1 (Data Preprocessing):** In this stage, we partition the road map into small grid cells and calculate the travel demands and traffic status (*i.e.*, taxi inflow) of all grid cells and time intervals.

• **Stage 2 (TrafficGAN Training):** In this stage, we train TrafficGAN, a novel generative model for traffic estimation. TrafficGAN automatically captures the shared patterns across spatial regions of how traffic distributions evolve according to travel demand changes and underlying road network structures.

• **Stage 3 (Urban Plan Evaluation):** In this stage, the generator obtained from Stage 2 will be used to estimate the future traffic distribution $\boldsymbol{M}_R(\hat{d}_R)$ for a target region R, given

[1] https://developers.google.com/maps/documentation/geocoding/
[2] http://www.openstreetmap.org/

a deployment plan with the expected travel demand $\hat{d}_R$. Depending on traffic distribution requirement defined by the urban planners/authorities, the deployment plan is accepted or rejected.

2.1.2 Related Work

Urban planning is a technical and political process concerning with urban data analysis, urban data mining, off-deployment evaluation and urban design. The off-deployment evaluation problem is a vital and difficult part among all the urban problems. Related works are summarized below.

Traffic volume prediction. Some previous works focus on traffic volume prediction from different perspectives. For example, [17] proposes a hybrid framework that integrates both state-of-art machine learning techniques and well-established traffic flow theory to estimate citywide traffic volume. In [18] and [19], the authors develop models to predict the road traffic volume and crowd flows in subway stations. These work assume unchanged urban settings and predict the traffic volume over time and locations. However, in this work we aim to generate the traffic distributions under various travel demands, which are significant changes to the urban settings.

Graph Convolutional Networks (GCN). [20] is usually used to classify the nodes in a graph. It can be seen as a generalization of neural network models like CNN to graphs and networks. GCN applies graph convolutional layers inside the model with a feature matrix and an adjacency matrix as inputs, where each row of the feature matrix contains the features of one node, and the adjacency matrix is a representative description of the graph structure. Even though GCN takes the graph structure and the correlation between two nodes into consideration in convolution process, it is not a generative model which is in need to solve the traffic estimation problem.

Deep Learning for Urban Computing. Urban Computing is a general research area

which integrates urban sensing, data management and data analysis as a unified process to explore, analyze and solve crucial problems related to people's everyday life [21, 22, 23, 24]. With the recent rapid development of deep learning techniques, many researchers have made attempts to use deep learning models to solve urban computing problems. For example, Yuan et al. [24] propose to use a variation of the ConvLSTM model to predict traffic accidents. Wu et al use recurrent neural networks (RNN) to predict trajectories. Huang et al. [22] employ a deep attention model to predict crimes. Li et al. [23] employ a reinforcement learning method to dynamically reposition shared bikes. These work, however, do not use a generative model and they are very different from our problem.

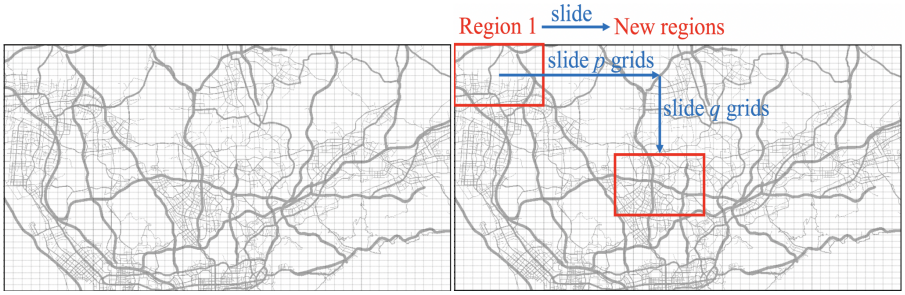
Other Generative Models have been discussed previously when motivating the TrafficGAN model. They do not capture irregular spatial structures of the road networks and the traffic correlations and thus could not effectively solve our problem.

2.1.3 Methodology

2.1.3.1 Stage 1: Data Preprocessing

Map Gridding For the ease of implementation in practice, we adopt the grid based method, which simply partitions the map into equal side-length grids [25, 26]. The grid based method has the advantage of allowing us to adjust the side-length of grids, which helps to examine and understand impacts of the grid size. In this paper, we divide the map of Shenzhen City into 40×50 grid cells with a side-length $l_1 = 0.0084^\circ$ in latitude and $l_2 = 0.0126^\circ$ in longitude.

Given all 40×50 grid cells in Shenzhen, we choose target region size $\ell = 10$ as an example in this study, where our TrafficGAN actually applies to any target region size. Thus, there are in total $1,271$ possible target regions with size 10×10. As shown in Fig. 2.5b, the upper-left red box is the first region in Shenzhen, to get new regions, we



(a) Shenzhen road map gridding (b) Regions illustration

Figure 2.5: Map gridding and regions illustration

can slide it horizontally for p grid cells, $0 \leq p \leq 40$, and/or vertically for q grid cells, $0 \leq q \leq 30, p, q \in \mathbb{N}$. The location of each region is described with a tuple (i, j) which indicates the coordinates of the first grid cell (the upper-left one) in the region, *i.e.*, the row and column index $0 \leq i \leq 30, 0 \leq j \leq 40, i, j \in \mathbb{N}$. However, it is unnecessary and too costly to use data from all $1,271$ regions as training data. Instead, we set $p = q = 5$ and get 63 regions covering entire Shenzhen city as target regions, extract their traffic distributions and travel demands over time.

Travel Demand. We use six months taxi GPS records of Shenzhen, China in 2016 to extract the travel demand of each grid cell and region in Shenzhen. In each time slot, *i.e.*, one hour, we count the total pickup events within each grid cell and each $\ell \times \ell$ region. Since in each GPS record, the passenger indicator value indicates whether a passenger is on board, which can be easily used to monitor the pickup information.

Traffic Distribution. Traffic distributions reflect the traffic status in a region, which can be quantified with many measures such as traffic speed, volume, inflow/outflow. Taking traffic inflow as an example, it is a crucial metric capturing the amount of arrivals in each grid cell. Since it is hard to obtain the total traffic inflow in a grid cell including all transport modes, in this paper, we use taxi inflow to represent the traffic inflow and many studies has proved its effectiveness [27, 28]. In each time slot of each day, we count all taxis which stay or arrive at each grid cell as the taxi inflow.

2.1.3.2 Stage 2: TrafficGAN Training

Taking an analogy, our "off-deployment traffic estimation" problem is similar as image generation problem, where the traffic distribution of a region can be viewed as a gray-scale "image", where the traffic status (*e.g.*, inflow) of each grid can be viewed as a "pixel" value. Thus, image generation approaches, such as GANs [29], sound a promising solution. However, the unique challenges of our problem prevent the state-of-the-art GAN models from solving it. In this section, we highlight the technical challenges of our problem, summarize the state-of-the-art generative models, and introduce our TrafficGAN for off-deployment traffic estimation problem.

Challenges. To solve the off-deployment traffic estimation problem, we aim to generate the traffic distributions with respect to various travel demands and the road network structures in the target region, which is a challenging task for the following reasons:

• Traffic correlations along road networks. In a target region R, the traffic of neighboring grids along the underlying road networks has strong correlations. Capturing such correlations is non-trivial since the correlation patterns are defined by the road network structures, which may have irregular shapes (rather than squares or rectangles).

• Conditioned Traffic Distribution Generation. The generated traffic distribution is meaningful only when conditioned on the given region R and the travel demand d_R. However, how to design a generative model that outputs the traffic distributions for a desired region and travel demand is challenging.

Preliminaries. Generative adversarial networks (a.k.a. GANs) [29] have been widely employed to many applications, including image, text generation, domain adaptation, *etc*. GAN includes a generator which generates a new data instance with input as a random code in a low dimensional space, and a discriminator which evaluates input data instances for authenticity.

Conditional GANs with deep convolutional layers (cDCGANs) [30] are composed of

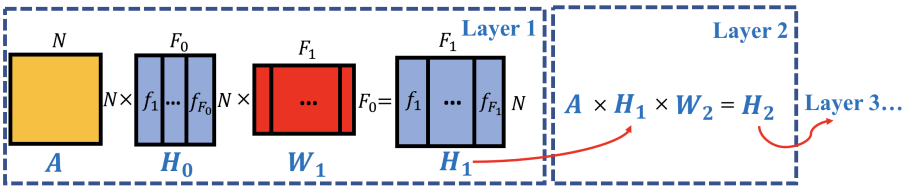


Figure 2.6: Propagation rule of dynamic convolutional layer, f refers to traffic features

multiple convolutional layers in both generator and discriminator to obtain better generation quality for primarily image data, and introduce a condition as input in both generator and discriminator to guarantee not only the generated data is close to the real, but also matches the input condition. cDCGAN seems a feasible method to solve the off-deployment traffic estimation problem since it can control the outputs by conditions and the convolutional layers can capture local patterns with filters. However, it is still hard to capture the traffic correlations along road networks accurately due to filters' fixed size and shape.

Below, we introduce a measure of traffic correlation across grid cells and develop TrafficGAN. As a generative model, TrafficGAN integrates the traffic correlations for traffic generations.

Quantifying Traffic Correlation. We introduce traffic correlation to capture the inherent traffic dependence between a grid cell pair. For each grid cell, there is time series traffic data (taxi inflow) over the entire study time period. We calculate the Pearson correlation coefficient between time series of a grid cell pair to quantify their traffic correlation. *Pearson correlation coefficient* [31] measures the linear correlation between variables X and Y, where X and Y are time series taxi inflow data of two grid cells in our case. The Pearson correlation coefficient a can be calculated by the formula below, where $\overline{X}$ and $\overline{Y}$ are the mean of X and Y:

$$a_{XY} = \frac{\sum_{i=1}^{n}\left(X_i - \overline{X}\right)\left(Y_i - \overline{Y}\right)}{\sqrt{\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2}\sqrt{\sum_{i=1}^{n}\left(Y_i - \overline{Y}\right)^2}}, \tag{2.1}$$

where $a \in [-1, 1]$. If $a \in [-1, 0)$, the two grid cells are negatively correlated, $a = 0$ means two grid cells don't have any linear correlation, $a \in (0, 1]$ indicates two cells are positively correlated. For an $\ell \times \ell$ region R, its corresponding traffic correlation matrix $\boldsymbol{A}$ is a symmetric $\ell^2 \times \ell^2$ matrix, where the entry a_{ij} is the traffic correlation between grid cell s_i and s_j, $s_i, s_j \in R$.

In a road network, nearby road segments (resp. nearby grid cells) often have stronger correlations in traffic according to the First Law of Geography [32]. In fact, the effective traffic correlations are generally positive, since if a road segment (resp. a grid cell) has traffic congestion, the road segments adjacent to it (resp. nearby grid cells) are likely to have high traffic volumes, the similar trend indicates positive correlations. However, the traffic correlations between distant road segments (resp. distant grid cells) are weak due to the lack of direct traffic connections. In our work, for a specific grid cell, we only consider its nearby grid cells which are directly connected with it by roads and thus (likely) to have positive traffic correlations. To remove other uncorrelated grids, we set a threshold $\lambda \in (0, 1)$ such that we set $a_{ij} = 0$, if $a_{ij} < \lambda$.

After removing the uncorrelated grids by the threshold λ, we perform row normalization for the traffic correlation matrix $\boldsymbol{A}$ as Eq. 2.2, so it will not affect the scale of features when multiplied to the feature matrix in Eq. 2.11.

$$a_{ij} = \frac{a_{ij}}{\sum_{j=1}^{\ell^2} a_{ij}}. \tag{2.2}$$

Next, we will elaborate on how to integrate the traffic correlation matrix for traffic distributions estimation and generation.

TrafficGAN. In this paper, to solve the challenges mentioned above, we propose a novel conditional generative model – TrafficGAN which can capture the traffic correlations of road networks, control the generation results with desired region and travel

(a) Filter in standard convolutional layer (b) Dynamic filter with threshold λ_1 (c) Dynamic filter with threshold λ_2

Figure 2.7: Convolutional filter comparison, $\lambda_1 > \lambda_2$

demand conditions, and generate realistic traffic distributions. TrafficGAN consists of a generator G and a discriminator D, and it applies dynamic convolutional layers in G and D.

The goal of dynamic convolutional layer is to learn a function of traffic features in a region including traffic inflow, volume, speed, *etc*. The input of dynamic convolutional layer includes two parts: • A traffic feature matrix $\boldsymbol{H}$ of size $N \times F_0$ (N: number of grid cells in a region, $N = \ell \times \ell$; F_0: initial number of traffic features).

• A non-negative and row-normalized traffic correlation matrix $\boldsymbol{A}$ of size $N \times N$.

The output is a new feature matrix after one-layer convolution. The layer-wise propagation rule is:

$$\boldsymbol{H}_{i+1} = f\left(\boldsymbol{H}_i, \boldsymbol{A}\right) = \sigma\left(\boldsymbol{A}\boldsymbol{H}_i\boldsymbol{W}_{i+1}\right), \tag{2.3}$$

where $\boldsymbol{H}_i$ is the feature matrix of a region got after ith layer and is the input of the $(i+1)$th layer, $\boldsymbol{W}_{i+1}$ is the weight matrix in $(i+1)$th layer and σ is an activation function. The rule is illustrated in Fig. 2.17.

Dynamic Filter. In the propagation, since the traffic correlation matrix is multiplied by the traffic feature matrix, for each grid cell, the new features after one-layer propagation is the weighted sum of all grid cells features within the corresponding region, and we can treat the correlations between the current grid cell and any other grid cells (*i.e.*, the corresponding row in correlation matrix) as a filter, whose shape and size are irregular and controlled by these correlations. Hence, we say such filters are dynamic since the the

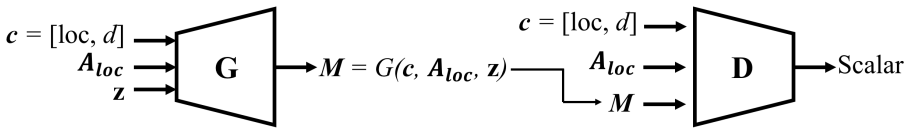


Figure 2.8: TrafficGAN

filter of each grid cell would be different and changeable.

Dynamic Convolutional Layer *vs.* Standard Convolutional Layer. Fig. 2.18 illustrates the difference between a standard convolutional layer and a dynamic convolutional layer (with different threshold λ). By introducing the traffic correlation matrix $\boldsymbol{A}$ in dynamic convolutional layer, a dynamic "filter" is created and applied to the feature matrix $\boldsymbol{H}$, where the size and the shape of the dynamic filter is controlled by $\boldsymbol{A}$ and the threshold λ, when λ changes, the dynamic filter for the same grid cell could be different. The filters are marked in yellow, and the blue block represents the target grid. The filter of a standard convolutional layer (Fig. 2.18(a) has fixed size, *e.g.*, a 3×3 square, which cannot naturally captures the traffic correlations along the road networks and would include some grids with no roads or some girds having low traffic correlations. In contrast, the dynamic filters created by the traffic correlation matrix (in Fig. 2.18(b)-2.18(c)) align with the road network very well. Moreover, comparing Fig. 2.18(b) and Fig. 2.18(c), it is clear that a smaller threshold λ leads to a larger range of dynamic filter, and vice versa.

Besides the dynamic filter, another filter $\boldsymbol{W}$ in Eq. 2.11 performs convolution on traffic features in each grid cell. Moreover, the corresponding dynamic de-convolutional layer is the same structure as the dynamic convolutional layer as shown in Fig 2.17. This is because the matrix operation of the dynamic convolutional layer and dynamic de-convolutional layer is invariant. We omit the detailed proof for brevity.

TrafficGAN Architecture. To tackle the challenge of conditioned traffic distribution generation, we introduce conditional generative model structure in designing TrafficGAN. Fig. 5.5 shows the overall structure of TrafficGAN. The goal of the generator G is to gen-

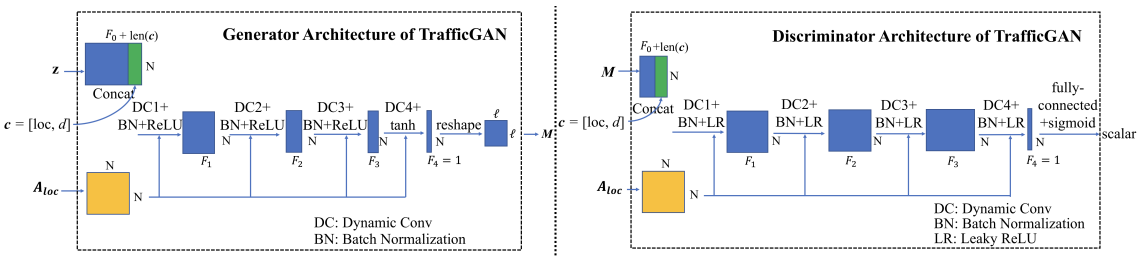


Figure 2.9: TrafficGAN architecture

erate traffic distributions with respect to the region location loc and travel demand d. The input of the generator G includes three parts, i) a low-dimensional code vector $\boldsymbol{z}$, randomly sampled from Gaussian distribution, ii) condition vector $\boldsymbol{c} = [\text{loc}, d]$, defining the desired region and travel demand and iii) a traffic correlation matrix $\boldsymbol{A}_{\text{loc}}$. The discriminator D takes three inputs, i) a traffic distribution $\boldsymbol{M}$, ii) condition information $\boldsymbol{c} = [\text{loc}, d]$ and iii) a traffic correlation matrix $\boldsymbol{A}_{\text{loc}}$. D outputs a scalar indicating whether the traffic distribution $\boldsymbol{M}$ is real and whether the input $\boldsymbol{M}$ and $\boldsymbol{c}$ are matched. The detailed structures of generator G and discriminator D are detailed in Fig. 2.9, len($\boldsymbol{c}$) represents the number of conditions in $\boldsymbol{c}$. In generator, $\boldsymbol{c}$ is concatenated into $\boldsymbol{z}$ such that the generator is conditioned by $\boldsymbol{c}$, which means the generator G builds the mapping from distribution $p_z(\boldsymbol{z})$ to a traffic distribution $G(\boldsymbol{c}, \boldsymbol{A}_{\text{loc}}, \boldsymbol{z})$. In our case, $N = 100$, $F_0 = 1$ since the only traffic feature is taxi inflow.

TrafficGAN Loss Function In TrafficGAN, the generator G aims to generate "like-real" traffic distributions so that the discriminator D cannot distinguish the generated traffic distributions from the real traffic distributions well. For the discriminator D, it aims to rise the score of real traffic distributions, lower down the score of generated traffic distributions, and lower down the score of mismatched pairs of traffic distributions and conditions. As a result, the loss function of TrafficGAN is in the form of Eq. 5.7, modeled

as a MinMax game. (See more details in [33].)

$$\min_G \max_D V(D,G) = E_{\boldsymbol{M}\sim p_{data}(\boldsymbol{M})}[\log D(\boldsymbol{c}, \boldsymbol{A}_{\text{loc}}, \boldsymbol{M})] + E_{z\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{c}, \boldsymbol{A}_{\text{loc}}, \boldsymbol{z})))]. \tag{2.4}$$

Input: Training iteration K, a training set $\mathcal{Z} = \{(\boldsymbol{c}^1, \boldsymbol{A}^1_{\text{loc}}, \boldsymbol{M}^1), \cdots, (\boldsymbol{c}^n, \boldsymbol{A}^n_{\text{loc}}, \boldsymbol{M}^n)\}$, initialized G and D.
Output: Well trained G and D.
1: In each training iteration $iter$:
2: **repeat**
3: Sample $\mathcal{Z}_0 = \{(\boldsymbol{c}^1, \boldsymbol{A}^1_{\text{loc}}, \boldsymbol{M}^1), \cdots, (\boldsymbol{c}^m, \boldsymbol{A}^m_{\text{loc}}, \boldsymbol{M}^m)\}$ from training set $\mathcal{Z}$, where $m < n$.
4: Sample $\mathcal{N} = \{\boldsymbol{z}^1, \boldsymbol{z}^2, \cdots, \boldsymbol{z}^m\}$ from Gaussian distribution.
5: Generate $\tilde{\mathcal{T}} = \{\tilde{\boldsymbol{M}}^1, \cdots, \tilde{\boldsymbol{M}}^m\}$ with G, where $\tilde{\boldsymbol{M}}^i = G(\boldsymbol{c}^i, \boldsymbol{A}^i_{\text{loc}}, \boldsymbol{z}^i)$.
6: Sample $\hat{\mathcal{T}} = \{\hat{\boldsymbol{M}}^1, \hat{\boldsymbol{M}}^2, \cdots, \hat{\boldsymbol{M}}^m\}$ from training set $\mathcal{Z}$, where each $\hat{\boldsymbol{M}}^i$ is mismatched with $(\boldsymbol{c}^i, \boldsymbol{A}^i_{\text{loc}})$.
7: Update D with Eq. 3.5 to maximize Eq. 2.15.
8: Update G with Eq. 3.6 to maximize Eq. 2.17.
9: **until** $iter > K$.

Algorithm 1: TrafficGAN Training Process

Training Process. During the training process, we apply batch gradient descent. The detailed training process is shown in Algorithm 9, where the discriminator D and the generator G are updated in line 3 – 7 and line 8, respectively.

In each training iteration, we update the parameters $\boldsymbol{\theta}_D$ of D with Eq. 2.15 and Eq. 3.5, where η_D is the learning rate.

$$\tilde{V}_D = \frac{1}{m}\sum_{i=1}^{m}\Big(\log(1 - D(\boldsymbol{c}^i, \boldsymbol{A}^i_{\text{loc}}, \tilde{\boldsymbol{M}}^i)) + \log D(\boldsymbol{c}^i, \boldsymbol{A}^i_{\text{loc}}, \boldsymbol{M}^i) + \log(1 - D(\boldsymbol{c}^i, \boldsymbol{A}^i_{\text{loc}}, \hat{\boldsymbol{M}}^i))\Big), \tag{2.5}$$

$$\boldsymbol{\theta}_D = \boldsymbol{\theta}_D + \eta_D \nabla \tilde{V}_{\boldsymbol{\theta}_D}(\boldsymbol{\theta}_D). \tag{2.6}$$

Then, we update the parameters $\boldsymbol{\theta}_G$ of G with Eq.2.17 and Eq.3.6, where η_G is the learning rate.

$$\tilde{V}_G = \frac{1}{m}\sum_{i=1}^{m} \log D(G(\boldsymbol{c}^i, \boldsymbol{A}^i_{\text{loc}}, \boldsymbol{z}^i)), \tag{2.7}$$

$$\boldsymbol{\theta}_G = \boldsymbol{\theta}_G + \eta_G \nabla \tilde{V}_{\boldsymbol{\theta}_G}(\boldsymbol{\theta}_G). \tag{2.8}$$

2.1.3.3 Stage 3: Urban Plan Evaluation

The generator G obtained from Stage 2 can be used by urban planners to evaluate urban construction plans at various locations, and search for more appropriate plans. To do so, given an urban deployment plan, the generator G takes (i) the expected travel demand $\hat{d}_R$, (ii) the location of the target region R, (iii) traffic correlation matrix of R, and (iv) random code vector $\boldsymbol{z}$, as inputs to generate traffic distributions for the plan to be evaluated.

Note that future traffic distributions hinge on many factors such as weather, *etc*. To capture the entire distribution of what the future traffic will look like over all potential (hidden) factors, we randomize a large number L of random code vectors to regenerate the traffic distributions for the urban plan. All L generated traffic distributions $[\tilde{\boldsymbol{M}}^1, \cdots, \tilde{\boldsymbol{M}}^L]$ are used to capture the future traffic distributions. The urban planners can summarize and evaluate various statistics of their interests using the L generated traffic distributions, for example, the mean, variance, minimum, maximum of L traffic distributions as outlined below.

Traffic mean distribution. The average of L generated traffic distributions reflects the average traffic status in the target region R after the plan is deployed. The positive or negative impacts of the urban construction plan on the local traffic status in R can be analyzed with the average generated distribution.

Traffic variance distribution. Similarly, we can take the variance of traffic status (*e.g.*, inflow) in each grid cell in R to obtain a traffic variance distribution, which indicates the

fluctuation of the generated traffic status in each grid cell.

2.1.4 Evaluation

We conduct experiments to evaluate our proposed TrafficGAN with baseline approaches using large scale real world taxi GPS data.

2.1.4.1 Experiment Design

We performed two sets of experiments: (i) Generate traffic distributions in a target region R that was "seen" by TrafficGAN in the training set but under other travel demands. To do this, we remove the traffic distribution $\boldsymbol{M}_R$ of R under a specific travel demand d_R from the training set, and train our model with the rest of the data. Then we let TrafficGAN generate the traffic distribution $\tilde{\boldsymbol{M}}_R$ under d_R and compare them with the removed $\boldsymbol{M}_R$.

(ii) Generate traffic distributions for an "unseen" target region R with a specific target travel demand d_R, where R was not included in the training set, therefore, TrafficGAN has never seen traffic distributions of R under any travel demand during training process. We extract the real traffic distributions of R from the original taxi GPS dataset and treat them as the ground truths, then we use the well-trained generator of TrafficGAN to generate the same number of traffic distributions and compare them with the ground truths. Obviously, the second task is more challenging.

In this paper, **Euclidean distance** and **mean absolute percentage error (MAPE)** are used to evaluate the quality of a generated traffic distribution against the ground truth traffic distribution of a target region R. Euclidean distance is defined as follows. For the ground-truth vector $\boldsymbol{V} = (v_1, \cdots, v_n)$ and the predicted vector $\hat{\boldsymbol{V}} = (\hat{v}_1, \cdots, \hat{v}_n)$, the

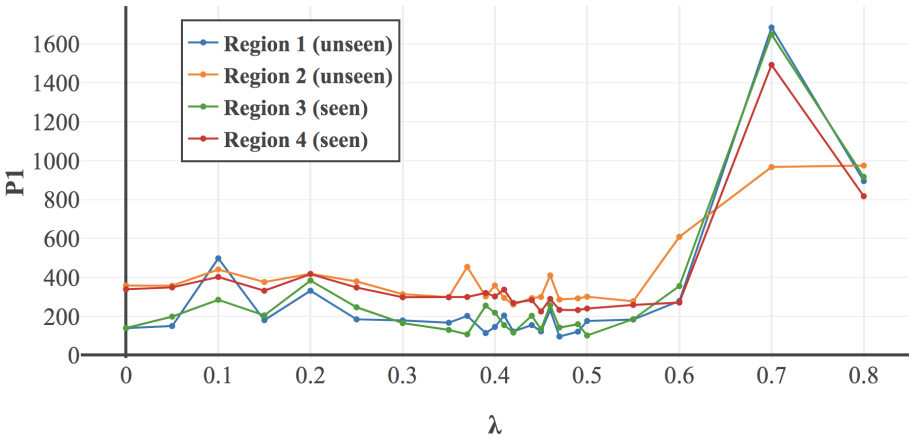


Figure 2.10: P_1 value changes with λ

Euclidean distance and MAPE between $\boldsymbol{V}$ and $\hat{\boldsymbol{V}}$ is:

$$\|\hat{V} - V\|_2 = \sqrt{\sum_{i=1}^{n} (\hat{v}_i - v_i)^2}. \tag{2.9}$$

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^{n} |v_i - \hat{v}_i| / v_i, \tag{2.10}$$

We define statistics P_1—P_4 (measured by Eq. 2.9), P_1'—P_4' (measured by Eq. 2.10) to measure and evaluate the difference between the generated traffic distribution and the ground-truth traffic distribution.

•$\boldsymbol{P_1}$ and $\boldsymbol{P_1'}$: For each R and d_R pair, we calculate the average traffic distribution using real traffic distributions and refer to it as "true average distribution". We also calculate the average of generated traffic distributions and refer to it as "generated average distribution". The "true average distribution" and "generated average distribution" can be reshaped into two vectors, the smaller Euclidean distance and MAPE between the two vectors (denoted with P_1 and P_1', respectively) reflect that the mean of the generated data are similar to the mean of the true data.

•$\boldsymbol{P_2}$, $\boldsymbol{P_3}$ and $\boldsymbol{P_2'}$, $\boldsymbol{P_3'}$: Under the condition of target region R and target travel demand d_R, for each grid cell $s \in R$, we calculate the Euclidean distance and MAPE between s in

Table 2.1: Statistics Comparisons for an "Unseen" Region

	TrafficGAN	**cGAN**	**cDCGAN**	**smoothing**	**regression**
P_1	**956.78**	14321.60	1452.82	1178.62	55302.89
P_1'	**3.55**	1710.41	3.71	21.27	220.79
P_2	**420.50**	7096.76	523.37	NA	NA
P_2'	**0.65**	253.48	1.25	NA	NA
P_3	**314.21**	1914.90	539.78	NA	NA
P_3'	**0.87**	400.30	3.09	NA	NA
P_4	**5249.24**	73505.63	7519.95	NA	NA
P_4'	**0.65**	253.48	1.25	NA	NA

Table 2.2: Statistics Comparisons for a "Seen" Region

	TrafficGAN	**cGAN**	**cDCGAN**	**smoothing**	**regression**
P_1	**896.95**	14436.03	1473.42	1418.74	57792.57
P_1'	**4.35**	1669.10	6.43	207.04	527.47
P_2	**361.74**	6393.57	455.25	NA	NA
P_2'	**0.73**	247.53	2.13	NA	NA
P_3	**277.67**	1837.80	512.83	NA	NA
P_3'	**0.89**	403.26	5.04	NA	NA
P_4	**4560.26**	66524.57	6857.47	NA	NA
P_4'	**0.73**	247.53	2.13	NA	NA

real traffic distributions and in generated distributions so that we have $N = \ell^2$ Euclidean distances and MAPEs for all $s \in R$. The mean of them (denoted as P_2 and P_2') indicate on average the similarity between the generated data and the true data for each grid cell. The standard deviation of these Euclidean distances and MAPEs are denoted as P_3 and P_3'.

• $\boldsymbol{P_4}$ and $\boldsymbol{P_4'}$ refer to the Euclidean distance and MAPE between real traffic distributions and generated traffic distributions with various travel demands. We combine all the real/generated traffic distributions with different travel demands as two huge matrices and reshape them into two vectors and calculate the Euclidean distance and MAPE between them, which indicate whether the traffic distributions conditioned on different travel demands are realistic or not.

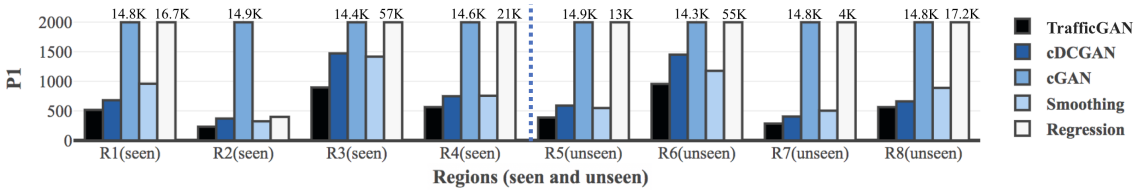


Figure 2.11: P_1 comparisons over 8 target regions (seen and unseen)

2.1.4.2 Baseline Models

We compare our TrafficGAN with four baseline approaches below.

Standard cGAN [33]. Without deep convolutional layers, the generator and discriminator are both composed of four fully-connected layers and the first three layers are activated by ReLU, output of the generator is activated by hyperbolic tangent function, and the output of discriminator is fed to Sigmoid function.

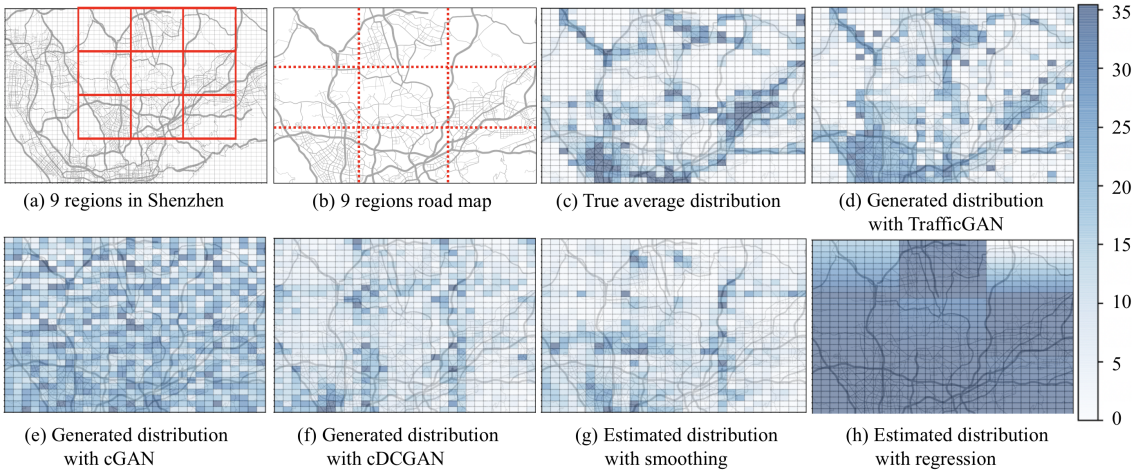
Conditional DCGAN [30]. The generator and discriminator of cDCGAN are composed of four transposed convolutional/convolutional layers and the first three layers are batch normalized and activated by leaky ReLU, the output of the generator is activated by hyperbolic tangent function, and the output of discriminator is activated by Sigmoid.

Spatial smoothing approach with neighboring regions[34]. This method uses the traffic distributions of 9 closest regions under the same travel demand to compute a mean distribution as the resulting estimation. Note we only use available data in the training set to estimate and we will ignore a neighboring region if its data is not available for this travel demand.

Regression [35]. Ridge regression is applied to estimate the taxi inflow of each grid cell with the location of the grid cell and the travel demand as predictors.

2.1.4.3 Experiment Settings

In the experiments, we obtain $122,472$ traffic distributions of Shenzhen, China from Jul 1st to Dec 31st in 2016. We train TrafficGAN, cGAN and cDCGAN both for 200 epochs,



(a) 9 regions in Shenzhen (b) 9 regions road map (c) True average distribution (d) Generated distribution with TrafficGAN

(e) Generated distribution with cGAN (f) Generated distribution with cDCGAN (g) Estimated distribution with smoothing (h) Estimated distribution with regression

Figure 2.12: Spatial patterns of 9 "unseen" regions

and randomly sample code $\boldsymbol{z}$ from a standard normal distribution with $\mu = 0, \sigma = 1$. All models are trained using Adam [11] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and a learning rate of 2×10^{-5} for the first 10 epochs and linearly decayed to 2×10^{-6}. In the training process, we use batch stochastic gradient descent with a batch size of 128.

2.1.4.4 Evaluation Results

λ selection. As we illustrated in Fig. 2.18, as a parameter of TrafficGAN, the correlation threshold λ would influence the performance of the generation. First we test the impact of different λ on the results. In Fig. 2.10, we pick two "seen" regions and two "unseen" regions with specific travel demands to see how their P_1 performances change with λ. We can see when λ moves from 0 to 0.4, the performance slightly improves (P_1 decreasing) but with big fluctuations. P_1 becomes more stable with smaller fluctuations when $0.4 < \lambda < 0.5$. When $\lambda > 0.6$, the P_1 increases drastically indicating bad performance. Based on our test when $\lambda = 0.47$, almost all regions have the lowest P_1. Thus, in this paper, we pick 0.47 as the proper value of λ, all the following experiments are conducted with $\lambda = 0.47$.

Statistics comparisons with four baselines. With $\lambda = 0.47$, we have the P_1 values

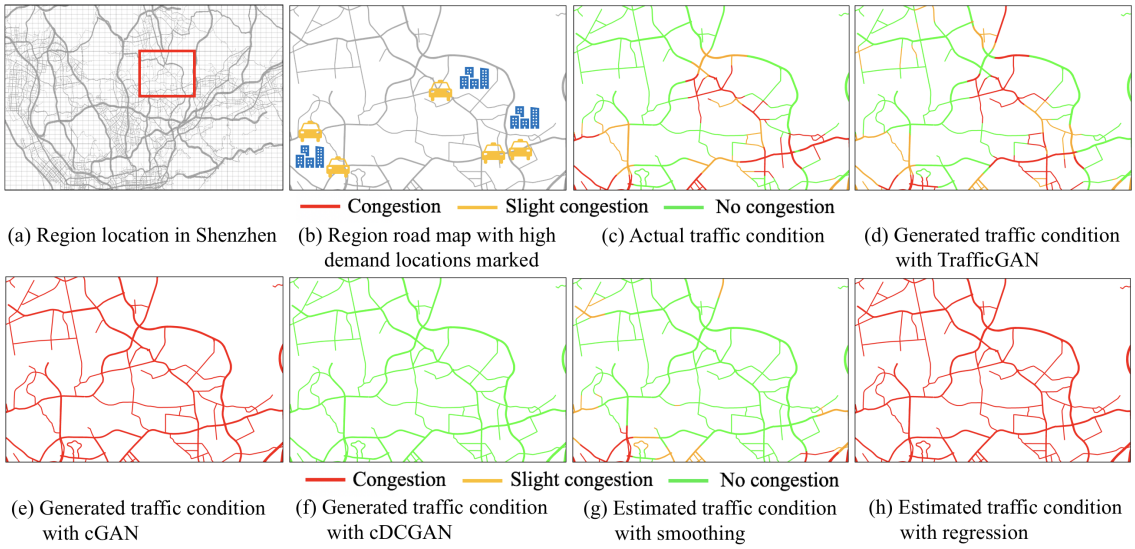


Figure 2.13: Traffic conditions of an "unseen" region

for 4 "seen" and 4 "unseen" regions in Fig. 2.11, where TrafficGAN always has the lowest P_1 indicating the mean of the generated data with TrafficGAN is the closest to the true data. Other statistics also have similar results.

We pick two representative regions (seen and unseen) as target regions with a specific travel demand. All the statistics are shown in Table. 2.1 and Table. 2.2. For both "seen" and "unseen" regions, TrafficGAN presents the lowest error in all statistics, which indicates the generated traffic distributions with TrafficGAN are much closer to the real ones. Compared with cGAN and cDCGAN, our TrafficGAN model brings down the P_1 error by up to 93.79% and 39.12% on the "seen" region and up to 93.32% and 34.14% on the "unseen" region.

Spatial pattern visualization. In this part, we visualize the generated/estimated traffic distributions and compare them with the real one. Here the traffic distributions are normalized to the same scale. Fig. 2.12 shows the visualizations of spatial patterns of 9 connected "unseen" regions, where each region has a corresponding travel demand. Fig. 2.12a marks the locations of selected 9 regions with red color on the whole city map. Fig. 2.12b shows the zoomed-in road map of the 9 regions. Fig. 2.12c shows the true

average distribution and Fig. 2.12d shows the generated average distribution with TrafficGAN. Fig. 2.12e - 2.12h show the generated/estimated average traffic distribution of the baselines. Obviously, the generated average distribution with TrafficGAN captures the structure of the underlying road networks of all 9 "unseen" regions. TrafficGAN clearly outperforms all the baselines which cannot accurately learn the spatial patterns of "unseen" regions and they usually overestimate or underestimate the value in each grid cell.

Traffic condition visualization. Moreover, in the traffic estimation problem, we focus more on estimating the traffic conditions in roads. Fig. 2.13 shows the traffic conditions in all roads in an "unseen" region under a specific travel demand, where the roads in red indicate congestion, the yellow roads indicates slight congestion, and the green ones represent no traffic congestion. Fig. 2.13a shows the location of this "unseen" region. Fig. 2.13b is the road map of the "unseen" region with high travel demand locations marked. Fig. 2.13c shows actual traffic condition in the data. Fig. 2.13d shows the generated traffic condition with TrafficGAN, which is highly similar to the ground truth. Fig. 2.13e - 2.13h show the results of the baselines. Clearly, the results of TrafficGAN outperforms all baselines. Results on the "seen" regions also suggest the same trend, where TrafficGAN can better capture the road networks and generate more realistic traffic distributions than all baselines. Due to space limit, we only present the results on "unseen" regions since it is a harder task.

In conclusion, TrafficGAN is a success in traffic estimation, which can not only capture the shared patterns across spatial regions of how traffic conditions evolve according to travel demand changes and underlying road network structures, but also provide realistic estimation of the traffic conditions in roads based on different travel demands.

(a) Longgang in Shenzhen (b) Current traffic condition (c) Forecast traffic condition (d) Possible reasons for (c)

Figure 2.14: Traffic condition forecast. (a) is the target region covering the Longgang District; (b) is the actual traffic condition with current travel demand in the region; (c) is the forecast traffic condition with a higher expected travel demand, where more congestion appears; (d) indicates two possible reasons for (c);

2.1.4.5 Case Studies

To further utilize our TrafficGAN, we look into real traffic condition evaluation cases in urban planning. As we mentioned earlier in this paper, the traffic condition always changes with the travel demand and rural areas usually have lower travel demands than urban areas. Therefore, it is a good opportunity to apply TrafficGAN in practice to forecast the possible traffic conditions under a not-yet-observed travel demand in an area.

For example, in 2018, Shenzhen government announced the plan to expropriate residential building. A large part of residential buildings to be expropriated are located in Longgang District. The goal of the expropriation is to build new residential and commercial areas in Longgang. The urban off-deployment traffic estimation can be performed before the expropriation and construction.

Longgang District is mainly located in the region marked with red box in Fig. 2.14a. The current average travel demand is 192. Fig. 2.14b shows the current traffic conditions. If new residential and business areas are built in Longgang, the travel demand would grow rapidly to 800 [16]. Fig. 2.14c shows the predicted traffic conditions of nearby roads under this expected travel demand. Compared with the current traffic conditions in Fig. 2.14b, apparently the overall traffic inflow is higher, the average traffic inflow increases drastically in two places marked in Fig. 2.14c. Fig. 2.14d illustrates possible reasons for the traffic congestion after the construction, *i.e.*, compacted roads and the

poor design of lanes in these areas.

2.2 Spatial-Temporal Generative Adversarial Networks

2.2.1 Overview

2.2.1.1 Introduction

The fast urbanization in recent years has brought huge impacts on urban traffic due to the growth of urban population, which potentially increases the travel demands and the risk of worsening traffic conditions caused by the overload of the transportation infrastructures. Therefore, urban traffic estimation has acted an important role in the process of urban development, which can provide insights for urban planning, traffic management and resource allocation, and help to improve the urban transportation efficiency and living environment [36]. For example, as shown in Figure 4.1, new sports village was planed to be built in Vaughan, Canada by the local government in 2019, which would increase the local travel demands to a great extent. Considering the potential traffic pressure the construction would bring, the plan was finally rejected [37]. Thus, urban traffic estimation is a critical step when evaluating an urban development plan before its deployment.

Given an urban development plan (with new travel demands it would produce), the underlying road network, and the historical traffic observations, *the problem of conditional urban traffic estimation* aims at evaluating the deployment plan by estimating traffic status under the new travel demands in consecutive time slots.

The conditional urban traffic estimation problem is challenging and difficult to solve due to the following reasons:

(1) *Traffic status heavily depends on travel demands.* Major changes in travel demands due to emergencies or urban constructions (*e.g.*, a newly constructed commercial center

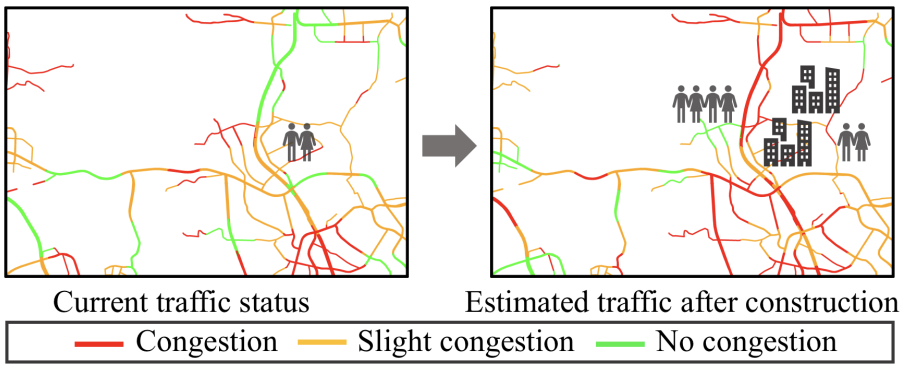


Figure 2.15: Example of traffic estimation and evaluation for urban planning in Vaughan, Canada.

or hospital) could drastically change the traffic status [1]. In such a scenario, data-driven approaches may not effectively estimate the traffic after the demand changes due to the lack of data.

(2) *Spatial auto-correlations.* The traffic status in nearby locations tends to correlate with each other. Capturing such auto-correlations is non-trivial. However, in a traffic network, the strength of traffic spatial auto-correlations varies at different locations and highly depends on the underlying road network structures.

(3) *Temporal auto-correlations.* Traffic status at the same location also exhibits strong auto-correlations over time. The traffic status at one location is highly correlated with its precedents. Such impacts also bring big challenges when estimating the urban traffic.

To estimate the urban traffic, many estimation methods have been proposed from different perspectives. The classic traffic estimation methods have been extensively studied in the literature [7, 8, 9, 10]. These works train machine learning models using historical traffic data trying to capture the correlations among the past traffic, environmental features and the future traffic. However, when predicting the traffic impacts of drastic increased (or decreased) travel demands, these models would fail because they cannot capture the future traffic changes caused by the travel demand changes due to the lack of training samples.

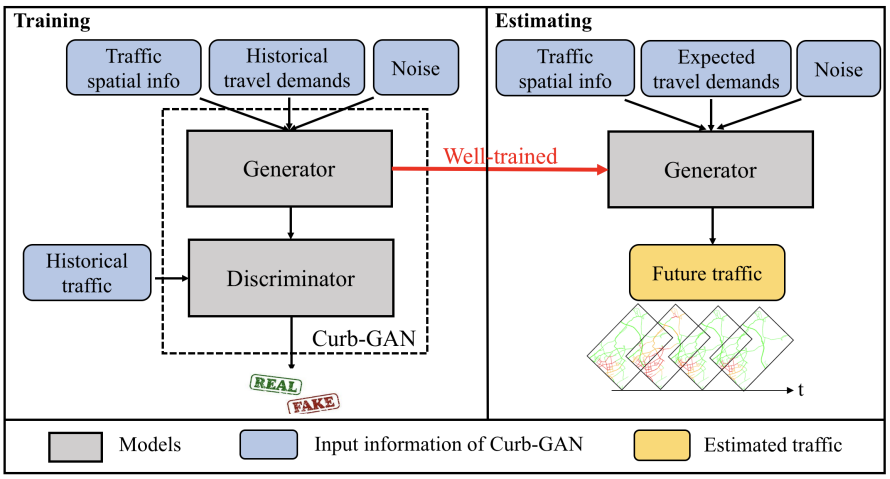


Figure 2.16: Insight of the framework.

Moreover, in recent years, there have been a lot of urban traffic prediction works using deep neural networks to model spatial and temporal auto-correlations. [38] used stacked autoencoders to predict the travel demands. [39] and [40] used ConvLSTM and ConvGRU to predict traffic accidents and crowd density. Others [41, 42] used the combination of CNN and LSTM to predict the road traffic speed and crowd flows. These models captured the temporal and spatial dependencies simultaneously, however, they did not consider the impact of conditions (*e.g.*, travel demand changes), cannot accurately capture the spatio-temporal auto-correlations with various travel demands, and thus fail to provide long-term predictions without any prior knowledge. A more recent work [1] proposed a TrafficGAN model to solve the traffic estimation problem. However, TrafficGAN ignores the temporal auto-correlations of traffic status and can only make snapshot estimations.

To tackle the aforementioned challenges and solve the conditional urban traffic estimation problem, in this paper, we propose a novel Conditional Urban Traffic Generative Adversarial Network (Curb-GAN), which can provide effective traffic estimations in consecutive time slots based on different travel demands. Figure 5.3 shows the solution framework, where the proposed Curb-GAN utilizes conditional GAN structure to control the generated traffic based on various travel demands, and the well-trained generator is

used to estimate future traffic. Curb-GAN features a few novel designs, including using dynamic convolutional layers to capture the spatial auto-correlations along the road networks, and applying self-attention mechanism to capture the traffic temporal dependencies across different time slots. Our **main contributions** are summarized as follows:

- We model the conditional traffic estimation problem as a traffic data generation problem, and propose a novel deep generative model Curb-GAN, which can generate future traffic estimations in consecutive time slots based on different travel demands in any region of a city.
- Building blocks containing dynamic convolutional layers and self-attention mechanism are designed to capture the shared patterns across spatio-temporal regions of how traffic status evolves according to time changes, travel demand changes and underlying road network structures.
- We conduct extensive experiments on two real-world spatio-temporal datasets (taxi inflow and traffic speed) to evaluate our proposed Curb-GAN. The experiment results verify that Curb-GAN can significantly improve the urban traffic estimation performance and outperform all existing baseline methods on both datasets.

2.2.1.2 Preliminaries

In this section, we first introduce the preliminaries used in this paper and then formalize the conditional urban traffic estimation problem.

Notations and Definitions We list the notations that will be used throughout the paper in Table. 5.1.

Definition 1 (Grid cells). We split the city into $I \times J$ grid cells with equal side-length in latitude and longitude, denoted as $\mathbb{S} = \{s_{ij}\}$, where $1 \leq i \leq I, 1 \leq j \leq J$.

Definition 2 (Target region). A target region R is a square geographic region in the city,

Table 2.3: Notations

Notations	Descriptions
i, j	Locations(coordinates in a grid world)
$\mathcal{S} = \{s_{ij}\}$	Grid cells
$\mathcal{R} = \{R_{ij}\}$	All target regions
$N_s = \ell^2 \in \mathbb{N}$	Number of grid cells in a region
$N_t \in \mathbb{N}$	Number of time slots within a day
$\tau_{\text{in}} \in \mathbb{N}$	Number of days of historical traffic observations
$\mathcal{D}^R = \{d_t^R\}$	Travel demand sequence of one day in R
$\mathcal{M}^R = \{\boldsymbol{M}_t^R\}$	Traffic distribution sequence of one day in R
$\mathcal{A}^R = \{\boldsymbol{A}_t^R\}$	Traffic correlation matrix sequence in R
$\mathcal{C}^R = \{\boldsymbol{C}_t^R\}$	Traffic condition sequence of one day in R

formed by $N_s = \ell \times \ell$ grid cells. The whole city can be split into overlapping regions $\mathcal{R} = \{R_{ij}\}$, where $R_{ij} = \langle s_{ij}, \ell \rangle$ is uniquely defined by an anchor grid cell s_{ij} on its top-left corner and a number ℓ of grid cells on the side[1].

Definition 3 (Travel demand). The travel demand of an area captures the total number of departures in a period of time. Thus, we denote the travel demand of a grid cell s in time slot t as $d_t^s \in \mathbb{N}$. Moreover, the travel demands of a target region R within a day is denoted as a sequence $\mathcal{D}^R = \{d_1^R, \ldots, d_{N_t}^R\} \in \mathbb{N}^{N_t}$, where d_t^R is the sum of travel demands in all grid cells within R in time slot t, *i.e.*, $d_t^R = \sum_{s \in R} d_t^s \in \mathbb{N}$.

Definition 4 (Traffic status and traffic distribution). Traffic status indicates the quality of traffic, which can be measured by traffic speed, traffic inflow/outflow, traffic volume, *etc*. We denote m_t^s as the average traffic status of grid cell s in time slot t. The traffic distributions of a target region R within a day is denoted as a tensor $\mathcal{M}^R = \{\boldsymbol{M}_1^R, \ldots, \boldsymbol{M}_{N_t}^R\} \in \mathbb{R}^{N_t \times \ell \times \ell}$, where we denote the $\ell \times \ell$ matrix $\boldsymbol{M}_t^R$ as the traffic distribution in R in time slot t, each entry of $\boldsymbol{M}_t^R$ is m_t^s, where $s \in R$.

Definition 5 (Traffic correlation matrix). The traffic correlation matrices of a target region R within a day is denoted as a tensor $\mathcal{A}^R = \{\boldsymbol{A}_1^R, \ldots, \boldsymbol{A}_{N_t}^R\} \in \mathbb{R}^{N_t \times N_s \times N_s}$, where $\boldsymbol{A}_t^R$ is a traffic correlation matrix of size $N_s \times N_s$ in region R in time slot t [1], $\boldsymbol{A}_t^R$ is

[1]Note that target regions can also be defined as rectangles rather than squares. For simplicity, we use square shape of target regions in this work.

non-negative and row-normalized.

Traffic correlations capture the inherent traffic dependencies between a grid cell pair (*i.e.*, auto-correlations). We use Pearson correlation coefficient of each pair of grid cells to quantify their corresponding traffic correlations. Note here we assume that traffic status at different locations are either positively correlated or independent. Negative correlations, though theoretically possible, are not considered in this paper.

Problem Definition A city area is partitioned into regions $\mathcal{R}$, given $\tau_{\text{in}} \times \|\mathcal{R}\|$ samples of $\mathcal{D}$ and $\mathcal{M}$, for one specific target region R, we aim to estimate the traffic distributions $\hat{\mathcal{M}}^R = \{\hat{\boldsymbol{M}}_1^R, \ldots, \hat{\boldsymbol{M}}_{N_t}^R\}$ in consecutive time slots based on a given expected travel demand sequence $\hat{\mathcal{D}}^R = \{\hat{d}_1^R, \ldots, \hat{d}_{N_t}^R\}$.

2.2.2 Related Work

Urban Traffic Prediction. Previous works focused on urban traffic prediction from different perspectives. There are some previously published works focusing on predicting an individual's movement based on their location history such as [43, 44]. They mainly forecast millions of individuals' mobility trajectories rather than the aggregated traffic conditions in a region. Some other researchers aimed to predict travel speed and traffic volume on roads. For example, [17] proposed a hybrid framework that integrated both state-of-art machine learning techniques and well-established traffic flow theory to estimate citywide traffic volume. In [18] and [19], the authors developed models to predict the road traffic volume and crowd flows in subway stations. These work assumed unchanged urban settings and predict the traffic volume over time and locations. Traditionally in civil engineering, agent-based simulation models [11] or physical models [12] were used to estimate the projected traffic volume after constructions. However, these models rely heavily on model choice and parameter settings, which are not transferable across urban regions. In our work, we focus on studying the spatio-temporal auto-correlations of traffic

and predict regional traffic based on different local travel demands.

Deep Learning for Spatio-Temporal Prediction. Deep learning methods have inspired many spatio-temporal applications. For example, CNNs were widely used in grid data modeling like citywide flow prediction [45] and taxi demand inference [46], since it can capture the spatial auto-correlations and thus provide good traffic estimations. Besides, RNNs [47] were also widely applied in spatio-temporal prediction problems due to their success in sequence learning. For example, [48] and [49] applied RNN to tackle video prediction and travel time estimation problem, respectively. In addition, [40] used ConvLSTM to to predict crowd density which captured temporal and spatial dependencies simultaneously and [41] used the combination of CNN and LSTM to predict the road traffic speed. Yuan et al. [39] proposed to use a variation of the ConvLSTM model to predict traffic accidents. Huang et al. [22] employed a deep attention model to predict crimes. Li et al. [23] employed a reinforcement learning method to dynamically reposition shared bikes. However, all these above studies are only trying to capture the spatial and temporal auto-correlations of traffic, they did not consider the impact of travel demand changes. In our study, we study the impact of travel demand changes and spatio-temporal auto-correlations simultaneously.

2.2.3 Methodology

To solve the conditional urban traffic estimation problem, we are inspired by the conditional GAN [33] model, since our traffic estimation problem is similar to conditional image sequences generation problem, where the travel demand can be treated as a condition, the traffic distribution of a region in one time slot can be treated as an“image” and the traffic status of each grid can be viewed as a “pixel” value. Thus, the conditional GAN (cGAN) structure could be potentially used to solve the conditional traffic estimation problem. However, the unique challenges (2) and (3) of our problem men-

tioned previously prevent the state-of-the-art cGAN models from solving it, since simple cGAN model cannot capture the spatial and temporal auto-correlations of traffic very well. Hence, we propose a novel generative model – Curb-GAN which can more accurately capture the spatial auto-correlations and temporal dependencies of traffic, control the generation results with desired travel demands, and generate realistic traffic estimations in consecutive time slots.

In this section, we introduce the architecture of Curb-GAN for traffic estimation problem. Following the conditional GAN structure, Curb-GAN consists of a generator G and a discriminator D, and both the generator and discriminator apply dynamic convolutional layers [1] and self-attention mechanism [50] to deal with the spatial and temporal auto-correlations of traffic.

2.2.3.1 Dynamic Convolutional Layer (DyConv)

In urban areas, the strength of traffic spatial auto-correlations is often heterogeneous, which mostly relies on the locations and the complex underlying road structures. Based on the First Law of Geography [32], nearby locations and closely connected roads often have stronger traffic spatial auto-correlations. Hence, we apply dynamic convolutional layers in both G and D, which can better capture the diverse footprints of spatial auto-correlations of traffic status.

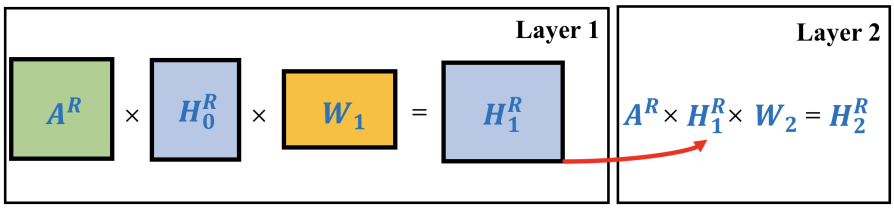
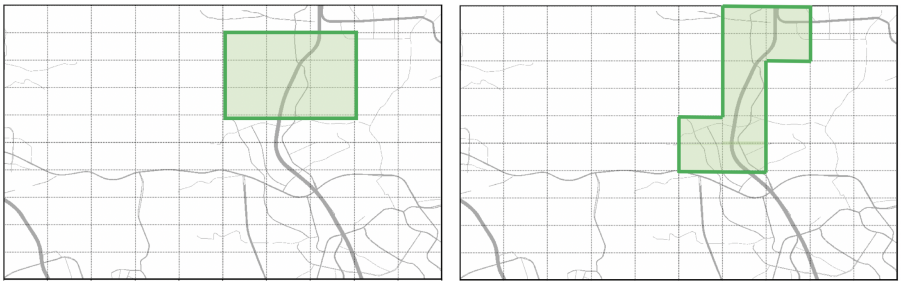


Figure 2.17: Propagation rule of DyConv.

The input of the dynamic convolutional layer includes: (1) a traffic status matrix $\boldsymbol{H}^R$ of size $N_s \times d_{\text{status}}$ (d_{status}: number of traffic status measures, if traffic status is represented



(a) Filter in standard Conv (b) Filter in DyConv

Figure 2.18: Filter comparisons of standard Conv and DyConv.

by only one measure, *e.g.*, traffic speed, then $d_{\text{status}} = 1$) and (2) a traffic correlation matrix $\boldsymbol{A}^R$.

The layer-wise propagation rule of DyConv is presented in Eq. 2.11 and the output of DyConv is a new traffic status matrix:

$$\boldsymbol{H}_i^R = f\left(\boldsymbol{H}_{i-1}^R, \boldsymbol{A}^R\right) = \sigma\left(\boldsymbol{A}^R \boldsymbol{H}_{i-1}^R \boldsymbol{W}_i\right), \tag{2.11}$$

where $\boldsymbol{H}_i^R$ is the output traffic status matrix of region R in i-th layer, $\boldsymbol{W}_i$ is the weight matrix and σ is an activation function. The rule is illustrated in Figure 2.17.

The traffic correlation matrix $\boldsymbol{A}^R$ in DyConv can also be viewed as a "filter", similar to the filter in a standard convolutional layer (Conv), which is applied to images and has fixed size and regular shape. The "filter" in DyConv created by $\boldsymbol{A}^R$ is applied to the traffic status matrix $\boldsymbol{H}^R$ which has irregular shape and size. As shown in Figure 2.18, the filter of standard Conv would cover some grids having no roads or very low traffic correlations and thus cannot capture the roads accurately, but the "filter" created by $\boldsymbol{A}^R$ in DyConv exactly captures the road structures since $\boldsymbol{A}^R$ can control the shape and size of the "filter" to make it only cover the grid cells which have very strong traffic correlations.

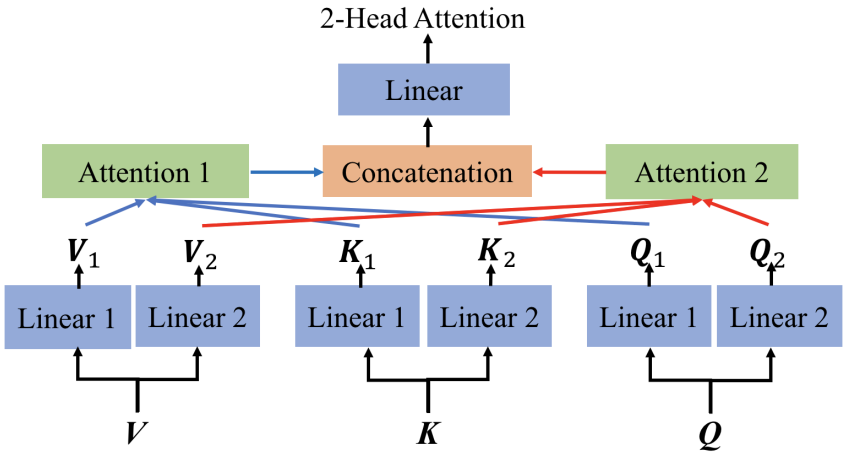
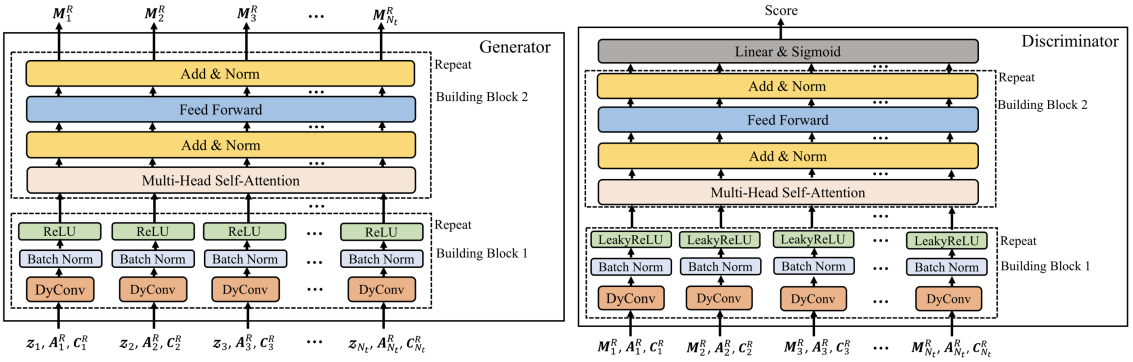


Figure 2.19: Example of 2 heads attention.

2.2.3.2 Self-Attention Mechanism (SA)

After applying the dynamic convolutional layer to capture the spatial auto-correlations of urban traffic, we are seeking a way to capture the temporal dependencies. Self-attention mechanism [50], which is mostly used in Seq2Seq models, achieves excellent performance when dealing with language modeling and machine translation problems. Self-attention mechanism handles sequential data including text, audios and videos, and learn the temporal dependencies from it. Compared with LSTM and GRU, self-attention mechanism is computed in parallel, and thus requires less time to train and results in higher training quality.

The input and output of a self-attention layer are two sequences of vectors. In the self-attention process, each vector in the input sequence is linearly transformed into three vectors called query, key and value. Each output vector is computed as a weighted sum of all the values, where the weights are the outputs of a softmax layer, and the inputs of the softmax layer are scaled dot products of the corresponding query with all keys. Since a sequence of queries, keys and values can be combined in matrices form $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$



(a) Generator of Curb-GAN

(b) Discriminator of Curb-GAN

Figure 2.20: Overview of Curb-GAN.

and computed in parallel, the self-attention function is calculated in Eq. 2.12.

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\boldsymbol{Q}\boldsymbol{K}^T/\sqrt{d_k}\right)\boldsymbol{V} \tag{2.12}$$

where d_k is the dimension of $\boldsymbol{K}$.

In this work, we apply multi-head self-attention mechanism, where the queries, keys and values are linearly transformed h times and thus we get h different attentions, which are then concatenated together and go through a linear transformation to get the final values, the multi-head self-attention is calculated with Eq. 2.13.

$$\begin{aligned} \text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= \text{Concat}\left(\text{head}_1, \ldots, \text{head}_\text{h}\right) W^O \\ \text{where head}_\text{i} &= \text{Attention}\left(\boldsymbol{Q}\boldsymbol{W}_i^{\boldsymbol{Q}}, \boldsymbol{K}\boldsymbol{W}_i^{\boldsymbol{K}}, \boldsymbol{V}\boldsymbol{W}_i^{\boldsymbol{V}}\right) \end{aligned} \tag{2.13}$$

where $\boldsymbol{W}_i^{\boldsymbol{Q}} \in \mathbb{R}^{d_\text{model} \times d_k}$, $\boldsymbol{W}_i^{\boldsymbol{K}} \in \mathbb{R}^{d_\text{model} \times d_k}$, $\boldsymbol{W}_i^{\boldsymbol{V}} \in \mathbb{R}^{d_\text{model} \times d_v}$ and $\boldsymbol{W}^{\boldsymbol{O}} \in \mathbb{R}^{hd_v \times d_\text{model}}$ are parameter matrices, d_v is the dimension of $\boldsymbol{V}$ and d_model is the dimension of the outputs. Figure 2.19 shows an example of 2-head attention.

2.2.3.3 Curb-GAN Architecture

To provide daily consecutive traffic estimations conditioned on expected travel demands, we employ the conditional GAN (cGAN) structure to make it possible to control the estimations by different travel demands. Figure 5.5 shows the overall structure of Curb-GAN. Curb-GAN contains a generator G and a discriminator D. The generator G aims to generate sequences of traffic distributions in consecutive time slots which are similar to the real ones so that the discriminator D cannot distinguish the generated traffic distribution sequences from the real sequences well.

The generator G aims to generate daily sequential traffic distributions with respect to the daily travel demand sequence $\mathcal{D}^R$ in a specific region. The input of the generator G includes three parts, i) a noise tensor $\mathcal{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{N_t}\} \in \mathbb{R}^{N_t \times N_s \times N_s}$, randomly sampled from Gaussian distribution, ii) a condition tensor $\mathcal{C}^R = \{\boldsymbol{C}_1^R, \ldots, \boldsymbol{C}_{N_t}^R\} \in \mathbb{R}^{N_t \times N_s \times 4}$, where $\boldsymbol{C}_t^R$ is a matrix of size $N_s \times 4$ defining the region location of R, travel demand and current time slot, *i.e.*, $\boldsymbol{C}_t^R = \text{Repeat}(\text{Concat}(i, j, d_t^R, t))$, where (i, j, d_t^R, t) are concatenated to one vector and repeat for N_s times to form the matrix $\boldsymbol{C}_t^R$, and iii) a traffic correlation matrix tensor $\mathcal{A}^R$. In generator, $\mathcal{C}^R$ is concatenated into $\mathcal{Z}$ and it builds the mapping from distribution $p_{\mathcal{Z}}(\mathcal{Z})$ to a traffic distribution $G(\mathcal{C}^R, \mathcal{A}^R, \mathcal{Z})$.

The discriminator D tries to rise the output score if the input is real traffic distribution sequence, and lower down the score if the input is generated traffic distribution sequence. D takes three inputs, i) a one day traffic distribution tensor $\mathcal{M}^R$, ii) a condition tensor $\mathcal{C}^R$ and iii) a traffic correlation matrix tensor $\mathcal{A}^R$. D outputs a scalar indicating whether the input traffic distribution tensor $\mathcal{M}^R$ is real and whether the input $\mathcal{M}^R$ and $\mathcal{C}^R$ are matched. The detailed structures of generator G and discriminator D are illustrated in Figure 5.5(a) and Figure 5.5(b).

As a result, the loss function of Curb-GAN is in the form of Eq. 5.7, modeled as a

Min-Max game with an additional L2 penalty. (See more details in [33].)

$$\min_G \max_D V(D,G) = E_{\mathcal{M}\sim p_{data}(\mathcal{M})}[\log D(\mathcal{C},\mathcal{A},\mathcal{M})] + E_{\mathcal{Z}\sim p_{\mathcal{Z}}(\mathcal{Z})}[\log(1-D(G(\mathcal{C},\mathcal{A},\mathcal{Z})))]. \tag{2.14}$$

Inside the generator and discriminator, we apply dynamic convolutional layer and self-attention mechanism, which help to capture the spatio-temporal auto-correlations. As shown in Figure 5.5, there are two building blocks inside G and D – Building Block 1 and Building Block 2, both can be stacked for several times.

Building Block 1 is composed of DyConvs followed by batch normalization and activation functions like ReLU or LeakyReLU. In Building Block 1, the number of DyConvs is equal to N_t, and all DyConvs can share the parameters.

Building Block 2 is composed of a multi-head self-attention layer and a feed-forward network composed of two fully-connected layers activated by ReLU. Both the self-attention layer and the feed-forward network are followed by an addition operation and a layer normalization [50].

Input: Training iteration k, a training set $\mathcal{P}$, initialized G and D.
Output: Well trained G and D.
1: In each training iteration $iter$:
2: **repeat**
3: Sample $\mathcal{P}_0$ from training set $\mathcal{P}$.
4: Sample $\mathcal{B}$ from Gaussian distribution.
5: Generate $\tilde{\mathcal{O}}$ with G.
6: Sample $\hat{\mathcal{O}}$ from training set $\mathcal{Z}$.
7: Update D with Eq. 3.5 to maximize Eq. 2.15.
8: Update G with Eq. 3.6 to maximize Eq. 2.17.
9: **until** $iter > k$.

Algorithm 2: Curb-GAN Training Process

2.2.3.4 Curb-GAN Training

During the training process, we apply BPTT (backpropagation through time). The detailed training process is shown in Algorithm 9, where the discriminator D and the generator G are updated in line 3 – 7 and line 8, respectively. Denote the training set which contains n samples as $\mathcal{P} = \{(\mathcal{C}^1, \mathcal{A}^1, \mathcal{M}^1), \cdots, (\mathcal{C}^n, \mathcal{A}^n, \mathcal{M}^n)\}$, Denote $\mathcal{P}_0 = \{(\mathcal{C}^1, \mathcal{A}^1, \mathcal{M}^1), \cdots, (\mathcal{C}^m, \mathcal{A}^m, \mathcal{M}^m)\}$ (line 3) as a subset of $\mathcal{P}$ containing m samples, where $m < n$. Denote $\mathcal{B} = \{\mathcal{Z}^1, \mathcal{Z}^2, \cdots, \mathcal{Z}^m\}$ as a set of m noise tensors sampled from Gaussian distribution (line 4), $\tilde{\mathcal{O}} = \{\tilde{\mathcal{M}}^1, \cdots, \tilde{\mathcal{M}}^m\}$ as a set of m traffic distribution tensors generated with G (line 5), where $\tilde{\mathcal{M}}^i = G(\mathcal{C}^i, \mathcal{A}^i, \mathcal{Z}^i)$. Denote $\hat{\mathcal{O}} = \{\hat{\mathcal{M}}^1, \hat{\mathcal{M}}^2, \cdots, \hat{\mathcal{M}}^m\}$ as a set of m traffic distribution tensors sampled from the training set $\mathcal{P}$ (line 6), each $\hat{\mathcal{M}}^i$ is mismatched with $(\mathcal{C}^i, \mathcal{A}^i)$. In each training iteration, we update the parameters $\boldsymbol{\theta}_D$ of D with Eq. 2.15 and Eq. 3.5, where η_D is the learning rate.

$$\tilde{V}_D = \frac{1}{m}\sum_{i=1}^{m}\Big(\log(1 - D(\mathcal{C}^i, \mathcal{A}^i, \tilde{\mathcal{M}}^i)) + \log D(\mathcal{C}^i, \mathcal{A}^i, \mathcal{M}^i) + \log(1 - D(\mathcal{C}^i, \mathcal{A}^i, \hat{\mathcal{M}}^i))\Big), \tag{2.15}$$

$$\boldsymbol{\theta}_D = \boldsymbol{\theta}_D + \eta_D \nabla \tilde{V}_{\boldsymbol{\theta}_D}(\boldsymbol{\theta}_D). \tag{2.16}$$

Then, we update the parameters $\boldsymbol{\theta}_G$ of G with Eq.2.17 and Eq.3.6, where η_G is the learning rate.

$$\tilde{V}_G = \frac{1}{m}\sum_{i=1}^{m}\log D(G(\mathcal{C}^i, \mathcal{A}^i, \mathcal{Z}^i)), \tag{2.17}$$

$$\boldsymbol{\theta}_G = \boldsymbol{\theta}_G + \eta_G \nabla \tilde{V}_{\boldsymbol{\theta}_G}(\boldsymbol{\theta}_G). \tag{2.18}$$

After training, we use the well-trained generator to generate the estimated traffic distributions in consecutive time slots of target regions with expected travel demand sequences.

2.2.4 Evaluation

In this section, We first describe the two real-world spatio-temporal datasets and then introduce baselines and the evaluation metrics. Finally, we present and analyze our experiment results in detail.

2.2.4.1 Dataset Descriptions

We validate the effectiveness of our model on two real-world data sets: (1) traffic speed and (2) taxi inflow.

- **Traffic speed.** The hourly average traffic speed is extracted from GPS records collected from taxis in Shenzhen, China from Jul 1st to Dec 31st, 2016. In this estimation task, we first partition Shenzhen City into 40×50 grid cells. The traffic status in each grid cell is measured by average traffic speed, and there are 4416 time slots (*i.e.*, one hour) over 6 months. Then for each time slot (*i.e.*, one hour), we obtain traffic distributions and travel demands of training regions, and use the daily traffic distribution sequences and travel demand sequences of training regions to train the model. The goal of this task is to estimate the traffic distribution sequence $\hat{\mathcal{M}}^R$ of a test region R conditioned on the expected travel demand sequence $\hat{\mathcal{D}}^R$.

- **Taxi inflow.** The taxi inflow data is collected from taxis in Shenzhen, China from July 1st to Dec. 31st, 2016. In each time slot (*i.e.*, one hour) of each day, the taxi inflow is the count of all taxis that stayed or arrived at each grid cell. In this estimation task, the entire Shenzhen City is partitioned into 40×50 grid cells, and the traffic status in each grid cell is measured by taxi inflow. With the knowledge of $\mathcal{D}$ and $\mathcal{M}$ for all training regions, for a specific test region R, given the expected travel demand sequence $\hat{\mathcal{D}}^R$, we aim at estimating the traffic distribution sequence $\hat{\mathcal{M}}^R$.

2.2.4.2 Baselines

- **Spatial smoothing with neighboring regions [34].** In each time slot, this method uses the traffic distributions of 9 closest regions under the same travel demand to compute a mean distribution as the resulting estimation. Note we only use available data in the training set to estimate and we will ignore a neighboring region if its data is not available for this travel demand.

- **ConvLSTM [51, 52].** This method uses conditional GAN structure, and applies ConvLSTM inside both generator and discriminator to provide sequential estimated traffic distributions.

- **FC-SA [50, 52].** This method uses conditional GAN structure, and applies stacked fully-connected layers and self-attention layers inside both generator and discriminator.

- **CNN-SA [52, 53].** This method uses conditional GAN structure and applies stacked standard convolutional layers and self-attention layers inside generator and discriminator.

- **FC-LSTM [52, 54].** This method uses conditional GAN structure and applies stacked fully-connected layers and multi-layer LSTM inside generator and discriminator.

- **CNN-LSTM [52, 55].** This method uses conditional GAN structure and applies stacked standard convolutional layers and multi-layer LSTM inside generator and discriminator.

- **DyConv-LSTM [1, 52].** This method uses conditional GAN structure and applies stacked dynamic convolutional layers and multi-layer LSTM inside generator and discriminator.

Table 2.4: Performance results on traffic speed estimation and taxi inflow estimation.

Methods		Smoothing	ConvLSTM	FC-SA	CNN-SA	FC-LSTM	CNN-LSTM	DyConv-LSTM	**Curb-GAN**
Traffic speed	RMSE	16.37	18.90	44.30	38.06	128.03	30.15	22.72	**13.34**
	MAPE	0.94	1.07	3.44	3.02	3.70	2.27	1.26	**0.76**
Taxi inflow	RMSE	37.71	38.73	40.30	38.54	41.11	38.20	37.33	**36.29**
	MAPE	27.56	10.43	79.75	52.25	36.92	62.02	16.88	**5.88**

Table 2.5: Variants of Curb-GAN evaluations.

Methods		2DyConv+1SA	3DyConv+1SA	4DyConv+1SA	4DyConv+2SA	4DyConv+3SA
Traffic speed	RMSE	219.10	19.43	17.73	20.67	**13.34**
	MAPE	5.44	1.09	0.91	0.95	**0.76**
Taxi inflow	RMSE	62.05	59.98	41.66	37.03	**36.29**
	MAPE	138.21	33.15	28.16	16.85	**5.88**

2.2.4.3 Evaluation Metrics

We use mean absolute percentage error (MAPE) and rooted mean square error (RMSE) to evaluate Curb-GAN:

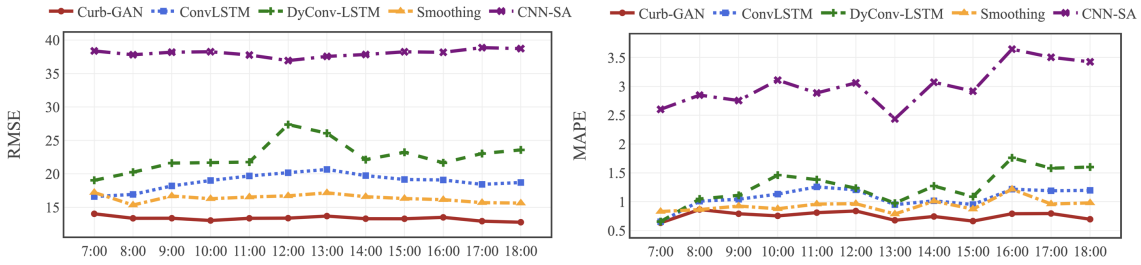
$$\text{MAPE} = \frac{1}{N_s N_t} \sum_{s=1}^{N_s} \sum_{t=1}^{N_t} \left| y_{s,t} - \hat{y}_{s,t} \right| / y_{s,t} \tag{2.19}$$

$$\text{RMSE} = \sqrt{\frac{1}{N_s N_t} \sum_{s=1}^{N_s} \sum_{t=1}^{N_t} \left(y_{s,t} - \hat{y}_{s,t} \right)^2} \tag{2.20}$$

where $y_{s,t}$ is the ground-truth traffic status observed in the s-th grid cell and t-th time slot, and $\hat{y}_{s,t}$ is the corresponding prediction.

2.2.4.4 Experimental Settings

The whole Shenzhen city is divided to 40×50 grid cells with a side-length $l_1 = 0.0084°$ in latitude and $l_2 = 0.0126°$ in longitude. Each region is of size 10×10, *i.e.*, $\ell = 10$ and $N_s = 100$. Thus, there are in total $1,271$ possible target regions with size 10×10. However, it is unnecessary and too costly to use data from all $1,271$ regions to train the model. Instead, we select 63 regions covering entire Shenzhen city as target regions for training, extract their traffic distributions and travel demands over time, and use the rest



(a) RMSE of selected models from 7:00 to 19:00

(b) MAPE of selected models from 7:00 to 19:00

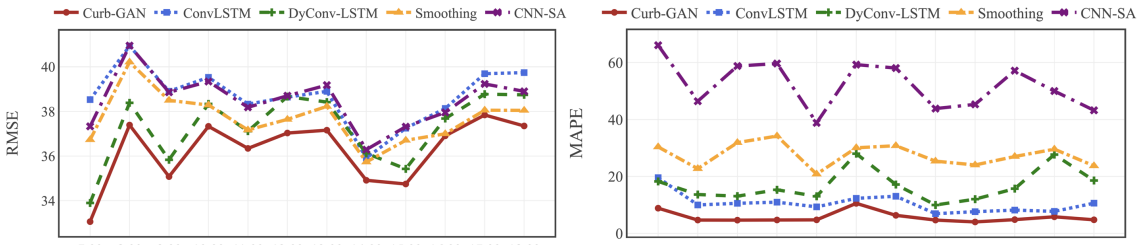
Figure 2.21: Comparisons of selected models in consecutive time slots in traffic speed estimation.

of regions for testing.

The daily time interval for the data used to train all the models are from 7:00am to 7:00pm, where each hour is a time slot and we have 12 time slots per day, *i.e.*, $N_t = 12$. Thus, the sequence lengths of $\mathcal{D}^R$, $\hat{\mathcal{D}^R}$, $\mathcal{M}^R$, $\hat{\mathcal{M}^R}$ and $\mathcal{A}^R$ are 12.

To extract the travel demands, in each time slot of a day, *i.e.*, one hour, we count the total taxi pickup events within each grid cell and each region. In general, it is hard to obtain the total travel demand in a region including all transport modes. In this work, we use the demand for taxis to represent the regional travel demand, where many studies have shown that taxi demands represent the total demands quite well [14, 15].

The structure of Curb-GAN is as follows: the Building Block 1 is stacked for 4 times, the Building Block 2 is staked 3 times, where 2-head self-attention is used. The initial input feature of building block 1 in generator is 104, the hidden features of stacked building block 1 are $\{64,32,16,1\}$. The initial input feature of building block 1 in discriminator is 5, the hidden features of stacked building block 1 are $\{16,32,64,1\}$. Curb-GAN is trained using Adam optimizer [56] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and a learning rate of 2×10^{-4} for 1500 epochs with a batch size of 64.



(a) RMSE of selected models from 7:00 to 19:00

(b) MAPE of selected models from 7:00 to 19:00

Figure 2.22: Comparisons of selected models in consecutive time slots in taxi inflow estimation.

2.2.4.5 Results

Average performance results. The performances of the competing baselines and Curb-GAN are shown in Table 5.2. For each dataset, we randomly pick one test region to calculate RMSE and MAPE with $N_t = 12, N_s = 100$, and similar results are got for other test regions. For deep models, we train and test each of them five times, the statistics are calculated using average generation results conditioned by the same travel demand sequence as the ground-truth.

In traffic speed estimation, Curb-GAN outperforms all the baselines on both metrics. Specifically, Curb-GAN shows 52.77% and 55.38% improvements on RMSE and MAPE beyond all baselines on average, respectively. Compared with FC-LSTM, FC-SA, CNN-LSTM and CNN-SA, Curb-GAN achieves significant improvements, because it explicitly models the relationships between different locations using DyConv. Smoothing seems to have low RMSE by simply averaging the traffic distributions of nearby regions, but it produces higher MAPE, which indicates bad spatio-temporal auto-correlations learned since the traffic of nearby regions cannot provide accurate estimates for the target region.

DyConv-LSTM simultaneously captures the spatial and temporal auto-correlations and it uses LSTM to handle the temporal dependencies, but its estimations are not as good as Curb-GAN due to the limitations of model expressiveness and non-parallel computations of LSTM and thus lead to more training time and lower generation quality.

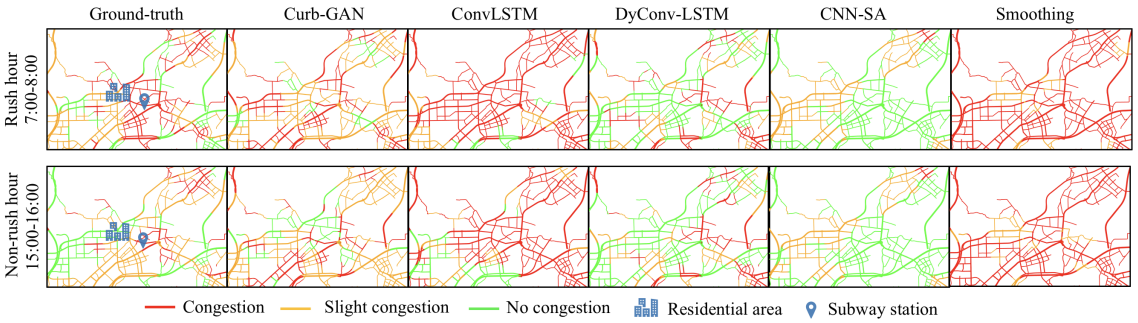


Figure 2.23: Traffic status visualizations.

In taxi inflow estimation, similar to traffic speed estimation, Curb-GAN significantly outperforms the baseline models by 6.48% and 77.63% improvements on average on RMSE and MAPE, respectively. The most competitive models are smoothing, ConvLSTM and DyConv-LSTM, but Curb-GAN can better learns the spatio-temporal patterns and thus obtain lower errors.

Performance in consecutive time slots. Since Curb-GAN is able to provide traffic estimations in consecutive time slots, to illustrate the effectiveness of Curb-GAN in traffic estimations in each time slot, we conduct experiments on Curb-GAN and four most competitive baseline models including ConvLSTM, DyConv-LSTM, CNN-SA and smoothing. The statistics are calculated in each time slot with $N_t = 1, N_s = 100$ using the average generation results of a specific test region based on the same travel demand sequence as the ground-truth.

In traffic speed estimation, as shown in Figure 2.21(a) and Figure 2.21(b), the Curb-GAN has the best performance in each hour from 7:00 to 19:00. In taxi inflow estimation, we got similar results in Figure 2.22(a) and Figure 2.22(b) that Curb-GAN outperforms the other four competitive baselines in both metrics from 7:00 to 19:00. These evaluations prove that the Curb-GAN can better capture the spatio-temporal auto-correlations and thus has a stable and excellent ability in estimating urban traffic in consecutive time slots.

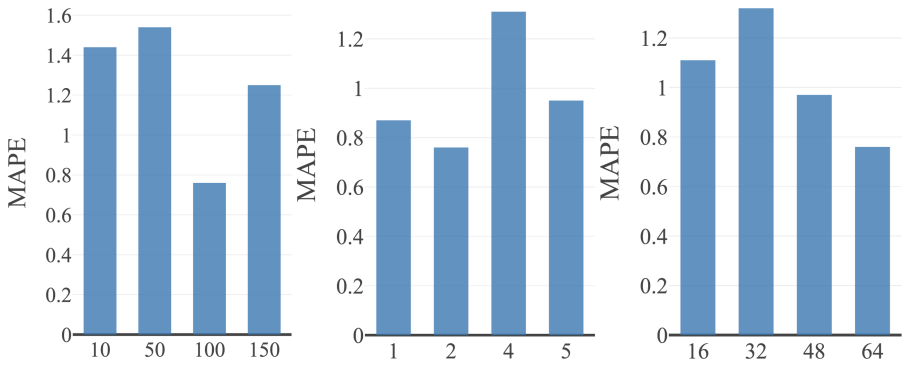
Traffic estimation visualizations To clearly validate the estimated traffic by Curb-

GAN against the ground-truth, we visualize the traffic distributions over the road networks. As shown in Figure 2.23, we pick two time slots of a day, *i.e.*, rush hour (7:00-8:00) and non-rush hour (15:00-16:00), and visualize the traffic status on the road map of a specific region in Shenzhen. The ground-truth visualizations are compared with the estimation visualizations of Curb-GAN and the other four competitive baseline models. Due to page limit, we only use traffic speed to measure the traffic status but we got similar results using taxi inflow.

In Figure 2.23, there are obvious traffic changes around the residential area and subway station (marked) between rush and non-rush hours in ground-truth visualizations. However, Smoothing, CNN-SA and DyConv-LSTM did not capture such spatio-temporal auto-correlations very well, as there is no traffic changes between the two hours, and the traffic status along the roads is different from the ground-truth. Even though smoothing and DyConv-LSTM got very competitive statistic results in Table 5.2 and Figure 2.21, they still cannot produce good estimations due to bad spatio-temporal patterns learned. ConvLSTM shows some traffic changes between two hours, but it produces worse spatial patterns compared with Curb-GAN. By contrast, our Curb-GAN generates reasonable traffic changes around residential area and subway station between rush and non-rush hours, which suggests that Curb-GAN can capture the spatial and temporal auto-correlations of traffic very well and produce more reliable traffic estimations then all baselines.

Evaluations on Curb-GAN parameters Curb-GAN has many settings including the number of stacked layers of Building Block 2 and Building Block2, initial dimension of noise, the number of heads in self-attention mechanism, *etc*. To investigate the robustness of Curb-GAN, we present the results under various parameter settings in both tasks.

Since the number of stacked Building Block 1 (DyConv layers) and Building Block 2 (self-attention layers) inside generator and discriminator could influence the final estima-



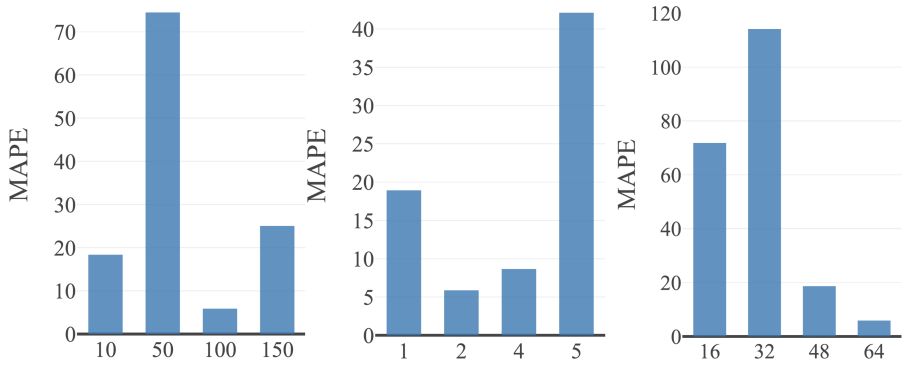
(a) Dimension of noise (b) Number of heads (c) Hidden features in DyConv

Figure 2.24: Impact of parameters in traffic speed estimation.

tion results, we evaluate the Curb-GAN with 2,3,4 stacked layers of Building Block 1 and 1,2,3 stacked layers of Building Block 2. The evaluations results are shown in Table 2.5. In both tasks, the more layers of Building Block 2 (self-attention) we use, the lower error we get in both metrics. It is because more layers of self-attention can better learned the temporal dependencies of traffic. When the number of DyConv layers increases from 2 to 4, the errors significantly decrease, which indicates too few of DyConv layers are not enough to capture the spatial auto-correlations, the structure of Curb-GAN should be adjusted to get the best estimation results for different datasets.

Next we test the impact of dimension of noise (See Figure 2.24(a) and Figure 2.25(a)), where the errors are both high when the noise dimension is too low or too high. With low dimension of noise, the fewer number of weights in DyConv is not enough to learn the spatial patterns, and it would need more time and training epochs to get good estimation results if the noise dimension is too high.

Then we test the impact of the head numbers in self-attention layers, Figure 2.24(b) and Figure 2.25(b) show the model performance with different number of heads and the same training epochs. The model with 2-head self-attention has better performance than the model with 1-head does, but with the number of heads increasing, the errors keep



(a) Dimension of noise (b) Number of heads (c) Hidden features in DyConv

Figure 2.25: Impact of parameters in taxi inflow estimation.

increasing. This indicates that more heads would record different temporal dependencies including local and long-term dependencies, but too many heads would weaken the capability of capturing the effective information and lead to higher errors.

We also change the number of hidden features in DyConv. The results of two tasks are shown in Figure 2.24(c) and Figure 2.25(c). Since we apply four layers of DyConv (four stacked Building Block 1) in traffic speed estimation, here we change the number of hidden feature of the first DyConv layer. We find that the model performance is sensitive to the hidden features, more hidden features in DyConv lead to better performance, which indicates more weights in DyConv can better capture the spatial patterns of traffic.

2.3 Complex-Condition-Controlled Urban Traffic Estimation

2.3.1 Overview

2.3.1.1 Introduction

Given the road network of a city and the historical traffic status (*e.g.*, volume, speed) over the network under various complex conditions (*e.g.*, travel demands, constructions, transit line designs) in the city, the problem of *conditional urban traffic estimation problem* aims at generating realistic traffic distribution projections of the city under new, previously unseen environmental conditions.

The urban traffic estimation problem has long been an important issue in various aspects of urban planning, including bus route planning, traffic management, land use design, *etc*. Accurate urban traffic estimation can not only help to reduce traffic congestion and improve the public transportation efficiency, but also provide insights for new urban constructions. For example, as illustrated in Figure 4.1, since the taxi demand greatly influenced the local traffic in Shenzhen, China, new subway stations were planned to be built to reduce the local taxi demand and thus release the traffic burden. Before the deployment, traffic estimations were conducted aiming to find the the best locations for new subway stations. Therefore, urban traffic estimation is an important step when evaluating an urban construction plan.

Challenges. Realistic and accurate urban traffic estimation is usually sophisticated and challenging due to the following reasons:

(1) *Complex conditions.* The urban conditions that affect traffic distributions are usually complex and unstructured in representation, such as multi-dimensional tensors or matrices (*e.g.*, subway routes) instead of simple labels or numeric measures. The complexity of

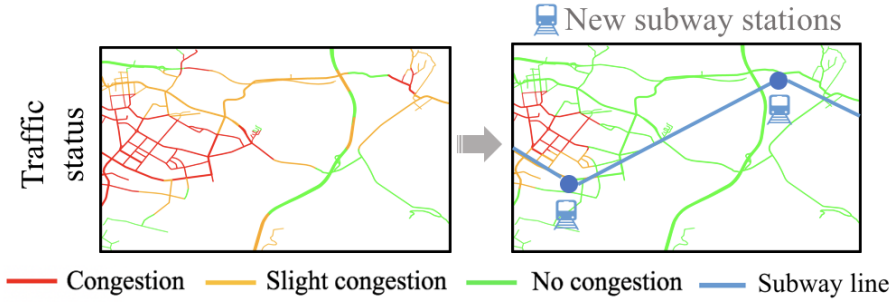


Figure 2.26: Traffic before & after building subway stations.

the conditions leads to difficulties in building a strong connection between the conditions and the traffic distribution, making it hard to capture the traffic changes caused by these factors.

(2) *Complex traffic spatial dependencies*[1]. The traffic status at a location is usually correlated with the traffic status in nearby locations. Such traffic dependencies are hard to capture since the underlying complex road networks usually lead to diverse traffic patterns.

The urban traffic estimation problem has received a lot of attentions in recent years. While most of the works addressed the second challenge above, *the first challenge is still unaddressed*. Some works [7, 8, 57] try to solve this problem with classical machine learning models. However, when estimating traffic status regarding to complex conditions, they typically cannot get good performance since they are incapable of dealing with complex conditions and accurately capturing traffic changes.

Recently, many works have focused on applying deep neural networks to solve traffic estimation problem. For example, stacked autoencoder [38] and ConvLSTM [39, 51] are used to predict travel demands and traffic accidents, respectively. These models greatly improve the prediction accuracy, however, they did not consider the impact of conditions and thus fail to solve the conditional urban traffic estimation problem. Besides, Traffic-

[1]In this paper, we focus on getting the estimation of the average traffic under specific conditions so do not consider the temporal dependencies here.

Table 2.6: Notations

Notations	Descriptions
i, j	Locations (coordinates in a grid world)
$\mathcal{S} = \{s_{ij}\}$	Grid cells within a city
$\boldsymbol{h} \in \mathbb{R}^{m \times n}$	Traffic condition
$\boldsymbol{x} \in \mathbb{R}^{m \times n}$	Traffic distribution
$\boldsymbol{z} \in \mathbb{R}^{v}$	Randomly sampled noise
$\boldsymbol{c} \in \mathbb{R}^{u}$	Embedded latent vector

GAN [1] and Curb-GAN [2] take simple conditions into account and estimate traffic with advanced GAN models. However, both of them cannot handle complex conditions, which usually lead to model collapse or instability.

Contributions. In this paper, we aim to solve the conditional urban traffic estimation problem and tackle both of the aforementioned challenges from a traffic data generation perspective. We propose a novel model — Complex-Condition-Controlled Generative Adversarial Network (C^3-GAN), which can successfully estimate traffic of an area based on complex urban conditions. Figure 4.2 is our solution framework. Our C^3-GAN features an embedding network and an inference network on top of the standard conditional GAN model, the well-trained embedding network and generator can be used for future traffic estimation. Our main contributions can be summarized as follows:

- We formulate the conditional urban traffic estimation problem as a traffic data generation problem, and propose a novel model — Complex-Condition-Controlled Generative Adversarial Network (C^3-GAN). C^3-GAN handles complex urban conditions through an embedding network which transforms the complex conditions to latent vectors, and an inference network which enhances the connections between the embedded vectors and the traffic data.

- We design a unique architecture for C^3-GAN to target the complex spatial dependencies challenge. C^3-GAN applies convolutional layers inside each model component to capture the traffic spatial dependencies, moreover, shared convolutional layers between

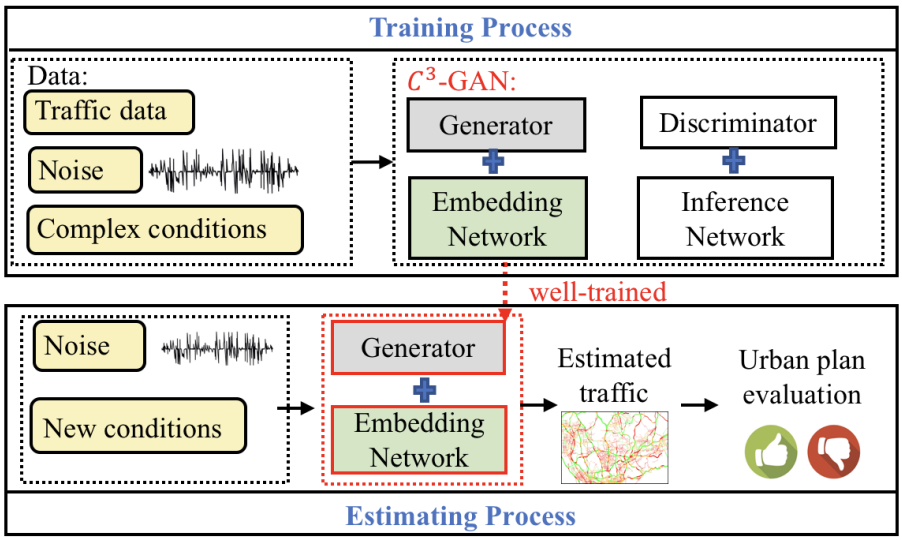


Figure 2.27: Solution framework.

the discriminator and the inference network help to capture spatial dependencies of traffic more efficiently, and thus get good performance. A novel training algorithm for C^3-GAN is also designed to guarantee the network stability, where part of the shared convolutional layers are used to update the embedding network periodically aiming to encourage good representation and avoid divergence.

- We perform extensive experiments on real-world datasets to evaluate our C^3-GAN. The experimental results prove that C^3-GAN can significantly improve the urban traffic estimation performance and outperform state-of-the-art baseline methods.

2.3.1.2 Preliminaries

The notations used in this paper are listed in Table 5.1. Next, we introduce the definitions and our problem statement.

Definition 1 (Grid cells). A whole city is divided into $m \times n$ grid cells, which have equal side-length in latitude and longitude. We denote the set of grid cells in the city as $\mathcal{S} = \{s_{ij}\}$, where $1 \leq i \leq m$ and $1 \leq j \leq n$.

Definition 2 (Urban conditions). Urban conditions (*e.g.*, travel demands, time of the day, *etc*) usually have strong correlations with road traffic. In this paper, we only consider complex urban conditions, *e.g.*, bus routes, rainfall intensity, *etc*. These complex conditions are usually presented in matrix form, thus, we denote a matrix $\boldsymbol{h} \in \mathbb{R}^{m \times n}$ as one feature map of the city in a period of time, where each element $h_s \in \mathbb{R}$ of the matrix indicates the corresponding condition in a specific grid cell $s \in \mathcal{S}$.

For example, if we use travel demand of a city as an urban condition, $\boldsymbol{h}$ will be a $m \times n$ travel demand matrix, each entry of the matrix indicates the average travel demand of a grid cell during a specific time slot.

Definition 3 (Traffic status and traffic distribution). Traffic status indicates the basic knowledge of the road traffic, which can be measured by different measurements, *e.g.*, traffic speed, traffic volume, *etc*. We denote x_s as the average traffic status of a grid cell $s \in \mathcal{S}$ within a period of time, and a matrix $\boldsymbol{x} \in \mathbb{R}^{m \times n}$ as the traffic distribution of the city.

Problem Statement: A city area is partitioned into grid cells $\mathcal{S}$, given historical samples of complex urban conditions $\mathcal{H} = \{\boldsymbol{h}_t\}$ and traffic distributions $\mathcal{X} = \{\boldsymbol{x}_t\}$ over a time span $1 \leq t \leq T$, we aim to estimate the future traffic distributions $\tilde{\boldsymbol{x}}$ given a set of new features $\tilde{\boldsymbol{h}}$.

2.3.2 Related Work

Now, we summarize the literature from two related areas, including urban traffic estimation, and generative adversarial networks.

Urban traffic estimation. There are many previous works focusing on urban traffic estimation. For example, a novel framework is proposed to predict traffic volume in the work [17] which combines classic machine learning techniques and well-established traffic flow theory, the work [18] utilizes location-based social media to predict the traffic

volume, and the work [19] proposes a real-time framework to predict crowd flows. All of these models are hard to get good performance when estimating traffic status regarding to complex conditions since they cannot deal with complex conditions very well.

Recently, deep learning has been successfully applied to various spatial-temporal prediction problems. For example, some works [45, 46] apply convolutional neural networks in citywide flow prediction and taxi demand inference, which help to better capture the spatial dependencies of data. Other works such as [49] and [41] try to apply recurrent neural networks [47] and LSTM [58] to tackle video prediction and travel time/speed prediction problem, since RNN and LSTM are good at learning long-term dependencies of data. Moreover, ConvLSTM [51] is also widely used in spatial-temporal prediction area, for example, the work [40] applies a variant of ConvLSTM to predict crowd density. However, all these works are not capable of taking complex conditions into account, some of them only consider simple conditions, the others do not include conditions in their models at all, thus, these existing models are improper to solve our conditional urban traffic estimation problem especially with complex conditions.

Generative adversarial networks. The core idea of the GAN model is to train the generator through the discriminator, where the discriminator is also being updated aiming to tell the real data from the fake data. GAN is widely applied in many areas, such as image generation, text-to-image translation, video prediction, *etc*. In recent years, a lot of GAN models are proposed. For example, CycleGAN [59] and StarGAN [60] are used for unsupervised image-to-image translation, cGAN [33] is the basis of other variants such as BiCoGAN [61] for supervised conditional generation, C-RNN-GAN [52] is designed for continuous sequential data generation. All these GAN models are successful in image generation, but when estimating traffic, most of them would fail since it is hard to capture the traffic dependencies and the correlations between traffic and various complex conditions at the same time. For traffic estimation, TrafficGAN [1] and Curb-GAN [2]

are proposed, but they work only when simple conditions are included, and thus cannot be used to estimate traffic based on complex conditions.

2.3.3 Methodology

Built upon the state-of-the-art (SOTA) literatures in generative models, we propose C^3-GAN for the conditional urban traffic estimation problem. C^3-GAN address the two challenges we mentioned previously with its unique designs:

(1) *Complex conditions challenge:* the proposed C^3-GAN introduces an embedding network and an inference network on top of the original cGAN to extract high-quality representations of the complex conditions and produce good generation results,

(2) *Complex traffic spatial dependencies challenge:* C^3-GAN applies convolutional layers inside each model component to capture the traffic spatial dependencies, moreover, shared convolutional layers between the discriminator and the inference network help to capture spatial dependencies of traffic more efficiently, and thus get good performance. Besides, to guarantee the model stability, a novel training algorithm for C^3-GAN is also designed.

2.3.3.1 SOTA of Deep Generative Models

Various generative adversarial networks (GANs) have been proposed to build mappings from simple distributions to data corpuses, *e.g.*, images, texts, *etc* [29, 52]. The general idea of conditional GANs matches the problem of urban traffic estimation very well. Below, we briefly introduce two GAN models that are relevant to our C^3-GAN, namely, the conditional GAN [33] and InfoGAN [62], and discuss the technical gaps for them to solve our problem.

Conditional GAN. The conditional generative adversarial network (cGAN) is a deep generative model proposed by Mirza *et al.* [33]. The generation process of cGAN is

governed by conditions, which tackles a min-max game as shown in Eq. 2.21. The goal of the generator G is to learn a distribution matching the real data distribution p_{data} using random noises $\boldsymbol{z} \sim p_{\boldsymbol{z}}$ and conditions $\boldsymbol{h}$, the discriminator D aims to distinguish the true data pairs from the generated ("fake") ones.

$$\begin{aligned}\min_G \max_D \mathcal{L}_{cGAN}(G, D) =& \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D(\boldsymbol{x}, \boldsymbol{h})] \\ &+ \mathbb{E}_{\boldsymbol{z}\sim p_z}[\log(1 - D(G(\boldsymbol{z}, \boldsymbol{h})))].\end{aligned} \tag{2.21}$$

Limitation of cGAN. Since our goal is to generate urban traffic estimations $\boldsymbol{x}$ using complex conditions $\boldsymbol{h}$ (*e.g.*, bus routes, travel demands), our intuition is to apply cGAN framework. However, cGAN usually deals with simple conditions (*e.g.*, discrete or continuous numbers), once the conditions become more complex (*e.g.*, multi-dimensional tensors or matrices), it is hard for standard cGAN to build strong connections between $\boldsymbol{x}$ and $\boldsymbol{h}$ and generate reasonable results due to the high-dimensionality of $\boldsymbol{h}$.

InfoGAN. InfoGAN proposed by Chen *et al.* [62] is an extension of GAN model [29], which adds a mutual information based regularizer to enable disentangled representations. To learn the semantic features of data, InfoGAN first splits the latent code into two parts — the disentangled code vector $\boldsymbol{c}$ and the remaining code vector $\boldsymbol{z}$, and then maximizes the mutual information $I(\boldsymbol{c}; G(\boldsymbol{c}, \boldsymbol{z}))$ and thus realize the goal of distinguishing data in an unsupervised fashion. The objective is given by the following expression:

$$\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D) - \lambda I(\boldsymbol{c}; G(\boldsymbol{c}, \boldsymbol{z})), \tag{2.22}$$

where $\mathcal{L}_{\text{GAN}}(G, D) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z})))]$.

Limitation of InfoGAN. Even though InfoGAN enables disentangled representations, it cannot be used to realize conditional traffic estimation with urban conditions (*e.g.*, bus routes, travel demands). InfoGAN learns data representations from unlabeled data in a

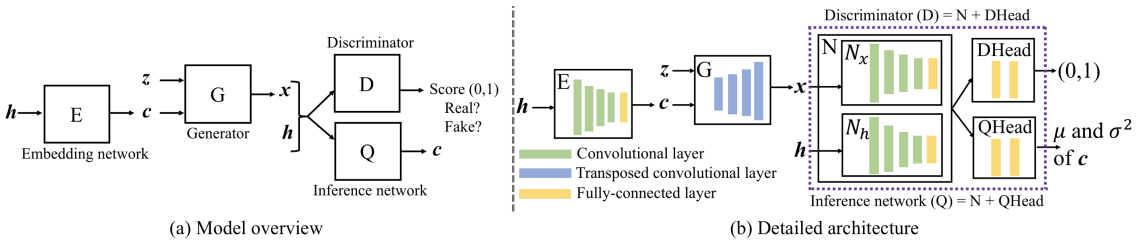


Figure 2.28: C^3-GAN structure.

unsupervised learning paradigm, which means the training data does not include corresponding conditions, thus, InfoGAN does not solve conditional generation problem and cannot be used to solve our problem. However, we are inspired by InfoGAN loss (especially the mutual information based regularizer) which can help to build strong connections between the generated data and the conditions.

2.3.3.2 Objective

Given the limitations of SOTA works of generative models to our traffic estimation problem, we propose a novel model C^3-GAN to tackle the conditional urban traffic estimation problem. The overview of C^3-GAN is shown in Figure 3.3(a). In C^3-GAN, we first focus on solving the complex condition challenge, thus, we propose to transform high-dimensional $\boldsymbol{h}$ to low-dimensional vector $\boldsymbol{c} \in \mathbb{R}^u$ with an embedding network E. The embedded latent vectors $\boldsymbol{c}$ should reflect key characteristics of the corresponding urban conditions $\boldsymbol{h}$. Once we use the embedded vector $\boldsymbol{c}$ and noise $\boldsymbol{z} \sim p_{\boldsymbol{z}}$ to generate the urban traffic $\boldsymbol{x} \sim G(\boldsymbol{z}, \boldsymbol{c})$, we need to ensure that the generated $\boldsymbol{x}$ is like real and matches the original condition $\boldsymbol{h}$, and also guarantee that the embedded latent vector $\boldsymbol{c}$ can accurately infer the corresponding traffic $\boldsymbol{x}$.

Since we use the embedding $\boldsymbol{c}$ of urban condition $\boldsymbol{h}$ for generation, the objective can

be written as:

$$\min_G \max_D V(G,D) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D(\boldsymbol{x},\boldsymbol{h})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1-D(G(\boldsymbol{z},\boldsymbol{c}),\boldsymbol{h}))]. \tag{2.23}$$

Eq.3.1 alone is certainly not good enough to produce good generation results, since the quality of $\boldsymbol{c}$ and the connections between $\boldsymbol{c}$ and $\boldsymbol{x}$ are not guaranteed. Borrowing the idea from InfoGAN, maximizing the mutual information $I(\boldsymbol{c};(G(\boldsymbol{z},\boldsymbol{c}),\boldsymbol{h}))$ can help to build strong connections between $\boldsymbol{c}$ and generated $\boldsymbol{x}$.

In information theory, mutual information between two random variables X and Y measures the "amount of information" learned from Y about X. The mutual information between X and Y and be written as:

$$I(X;Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X). \tag{2.24}$$

Based on Eq.2.24, the mutual information $I(X;Y)$ can be interpreted as the reduction of uncertainty in X when Y is provided. $I(X;Y) = 0$ represents X and Y are independent and knowing one variable reveals nothing about the other; by contrast, maximizing $I(X;Y)$ means Y can provide the most information about X.

Hence, to enhance the connections between embedded vectors $\boldsymbol{c}$ and the generated traffic $\boldsymbol{x}$, we propose to maximize $I(\boldsymbol{c};(G(\boldsymbol{z},\boldsymbol{c}),\boldsymbol{h}))$ instead of $I(\boldsymbol{c};G(\boldsymbol{z},\boldsymbol{c}))$, since both of $\boldsymbol{x} \sim G(\boldsymbol{z},\boldsymbol{c})$ and $\boldsymbol{h}$ contain the information of $\boldsymbol{c}$, which indicates $G(\boldsymbol{z},\boldsymbol{c})$ and $\boldsymbol{h}$ together have stronger mutual information with $\boldsymbol{c}$ than $G(\boldsymbol{z},\boldsymbol{c})$ alone does. Thus, maximizing $I(\boldsymbol{c};(G(\boldsymbol{z},\boldsymbol{c}),\boldsymbol{h}))$ can not only enhance the control of $\boldsymbol{c}$ over generated $\boldsymbol{x} \sim G(\boldsymbol{z},\boldsymbol{c})$ but also potentially accelerate the convergence. Therefore, we add a mutual information

regularizer to Eq.3.1:

$$\begin{aligned} &\min_G \max_D V_I(G, D) = V(G, D) - \lambda I(\boldsymbol{c}; (G(\boldsymbol{z}, \boldsymbol{c}), \boldsymbol{h})); \\ &\text{where } \boldsymbol{c} = E(\boldsymbol{h}). \end{aligned} \tag{2.25}$$

In practice, the mutual information term $I(\boldsymbol{c}; (G(\boldsymbol{z}, \boldsymbol{c}), \boldsymbol{h}))$ is hard to characterize analytically, since we do not have the access to the posterior distribution $P(\boldsymbol{c}|(G(\boldsymbol{z}, \boldsymbol{c}), \boldsymbol{h}))$. Instead, we can calculate the lower bound of $I(\boldsymbol{c}; (G(\boldsymbol{z}, \boldsymbol{c}), \boldsymbol{h}))$ and use the an auxiliary distribution $Q(\boldsymbol{c}|(G(\boldsymbol{z}, \boldsymbol{c}), \boldsymbol{h}))$ to approximate $P(\boldsymbol{c}|(G(\boldsymbol{z}, \boldsymbol{c}), \boldsymbol{h}))$. We denote $\hat{\boldsymbol{x}} = (G(\boldsymbol{z}, \boldsymbol{c}), \boldsymbol{h})$ for simplicity, the lower bound of $I(\boldsymbol{c}; (G(\boldsymbol{z}, \boldsymbol{c}), \boldsymbol{h}))$ is as follows:

$$\begin{aligned} &I(\boldsymbol{c}; (G(\boldsymbol{z}, \boldsymbol{c}), \boldsymbol{h})) = H(\boldsymbol{c}) - H(\boldsymbol{c} \mid (G(\boldsymbol{z}, \boldsymbol{c}), \boldsymbol{h})) \\ &= \mathbb{E}_{\boldsymbol{x} \sim G(\boldsymbol{z}, \boldsymbol{c}), \boldsymbol{h} \sim p_{\text{data}}} \left[\mathbb{E}_{\boldsymbol{c}' \sim P(\boldsymbol{c}|\hat{\boldsymbol{x}})} \left[\log P\left(\boldsymbol{c}' \mid \hat{\boldsymbol{x}}\right) \right] \right] + H(\boldsymbol{c}) \\ &= \mathbb{E}_{\boldsymbol{x} \sim G(\boldsymbol{z}, \boldsymbol{c}), \boldsymbol{h} \sim p_{\text{data}}} [\underbrace{D_{\mathrm{KL}}(P(\cdot \mid \hat{\boldsymbol{x}}) \| Q(\cdot \mid \hat{\boldsymbol{x}}))}_{\geq 0} \\ &+ \mathbb{E}_{\boldsymbol{c}' \sim P(\boldsymbol{c}|\hat{\boldsymbol{x}})} \left[\log Q\left(\boldsymbol{c}' \mid \hat{\boldsymbol{x}}\right) \right]] + H(\boldsymbol{c}) \\ &\geq \mathbb{E}_{\boldsymbol{x} \sim G(\boldsymbol{z}, \boldsymbol{c}), \boldsymbol{h} \sim p_{\text{data}}} \left[\mathbb{E}_{\boldsymbol{c}' \sim P(\boldsymbol{c}|\hat{\boldsymbol{x}})} \left[\log Q\left(\boldsymbol{c}' \mid \hat{\boldsymbol{x}}\right) \right] \right] + H(\boldsymbol{c}) \\ &= L_I(G, Q), \end{aligned} \tag{2.26}$$

where Q is the auxiliary distribution, and we can treat Q as a inference neural network which uses $\hat{x}$ to infer c just as illustrated in Figure 3.3(a). Moreover, we can simply omit $H(\boldsymbol{c})$ in $L_I(G, Q)$ since it is a constant when $\boldsymbol{c}$ is sampled from a fixed distribution. As a result, our final objective is as Eq.2.27:

$$\begin{aligned} &\min_{G,Q} \max_D V(D, G, Q) = V(G, D) - \lambda L_I(G, Q); \\ &\text{where } \boldsymbol{c} = E(\boldsymbol{h}). \end{aligned} \tag{2.27}$$

2.3.3.3 Practical challenges of Embedding Network

In our final objective Eq.2.27, a good fixed embedding network E is required, and three networks including G, D and Q are jointly trained. However, in practice, finding a good embedding is very challenging [63]. To mitigate this challenge, here are two naive approaches to obtain the embedding network:

(1) We can simply apply a randomly chosen embedding network for the purpose of transforming complex conditions $\boldsymbol{h}$ to simple embeddings $\boldsymbol{c}$. This method obviously doesn't present good performance, since a random embedding network cannot successfully extract features and usually lead to bad performance.

(2) The embedding network E can be jointly trained with the whole network. The objective function with a trainable E is as follows:

$$\begin{aligned}
&\min_{G,Q,E}\max_{D} V(D,G,Q,E) \\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[\log D(\boldsymbol{x},\boldsymbol{h})\right] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1-D(G(\boldsymbol{z},E(\boldsymbol{h})),\boldsymbol{h}))] \\
&- \lambda\mathbb{E}_{\boldsymbol{x}\sim G(\boldsymbol{z},\boldsymbol{c}),\boldsymbol{h}\sim p_{\text{data}}}\left[\mathbb{E}_{\boldsymbol{c}'\sim P(E(\boldsymbol{h})|\hat{\boldsymbol{x}})}\left[\log Q\left(\boldsymbol{c}' \mid \hat{\boldsymbol{x}}\right)\right]\right] \\
&- \lambda H(E(\boldsymbol{h})).
\end{aligned} \tag{2.28}$$

However, the whole training process may oscillate and fail to converge, namely, the posterior distribution $P(E(\boldsymbol{h})|\hat{\boldsymbol{x}})$ keeps changing with the evolving $E(\boldsymbol{h})$, which makes Q network have non-stationary targets and finally results in divergence. In addition, since E keeps training, $H(E(\boldsymbol{h}))$ cannot be ignored and will be maximized, which eventually results in an embedded distribution with high entropy.

Therefore, to get a good embedding network and avoid the problems mentioned above, we propose a uniquely designed architecture and training algorithm for our C^3-GAN. Next, we will introduce the detailed architecture and training algorithm of our model and

explain why they can help guarantee generation quality and model stability.

2.3.3.4 C^3-GAN Architecture

To tackle the complex spatial dependencies challenge, we design a unique architecture for C^3-GAN. Figure 3.3(b) shows the detailed architecture of C^3-GAN, which contains an embedding network E, a generator G, a discriminator D and an inference network Q. C^3-GAN applies convolutional layers inside each model component to capture the traffic spatial dependencies, moreover, shared convolutional layers between the discriminator and the inference network help to capture spatial dependencies of traffic more efficiently, and thus get good performance. Next, we will introduce our model architecture in detail.

The embedding network E aims to find a good latent representation for the high-dimensional urban condition $\boldsymbol{h}$. The input of E is an original condition $\boldsymbol{h}$, the output is a low-dimensional latent vector $\boldsymbol{c}$. Inside E, we have several convolutional layers, which help to capture the spatial patterns of urban conditions, each one is followed by a batch normalization and activated by Leaky ReLU [64], the final fully-connected layer is activated by hyperbolic tangent function.

The generator G aims to generate like-real traffic distributions with respect to an embedded latent vector $\boldsymbol{c}$. The input of the generator G includes i) a noise vector $\boldsymbol{z}$, which is randomly sampled from Gaussian distribution, *i.e.*, $\boldsymbol{z} \sim p_{\boldsymbol{z}}$, and ii) an embedded latent vector $\boldsymbol{c}$. G outputs the generated traffic distribution $\boldsymbol{x} \sim G(\boldsymbol{z}, \boldsymbol{c})$. Inside the generator G, $\boldsymbol{c}$ and $\boldsymbol{z}$ are concatenated together and go through some convolutional layers, where all the layers but the last one are batch normalized and activated by ReLU, the last layer is activated by hyperbolic tangent.

The discriminator D aims to distinguish the real data from the generated data by giving high score if the input $\boldsymbol{x}$ is from real data and matches the urban condition $\boldsymbol{h}$. It yields a low score if the input $\boldsymbol{x}$ is "fake" or does not match the input $\boldsymbol{h}$. Its input includes

i) traffic distribution $\boldsymbol{x}$, which can be real data sampled from the dataset or fake data generated by the generator, *i.e.*, $\boldsymbol{x} \sim p_{\text{data}}$ or $\boldsymbol{x} \sim G(\boldsymbol{z}, \boldsymbol{c})$, and ii) the urban condition $\boldsymbol{h}$.

The inference network Q aims to recover the distribution of latent vector $\boldsymbol{c}$ using $(\boldsymbol{x}, \boldsymbol{h})$ pairs, so Q takes the same input as D, which includes $\boldsymbol{x}$ and $\boldsymbol{h}$. In our design, Q and D share all convolutional layers and have separated fully-connected final layers, the shared convolutional layers between the Q and D can capture spatial patterns of traffic more efficiently and thus improve the training efficiency. As shown in Figure 3.3(b), we denote the shared part as N, the unique linear transformations for Q network is denoted as $QHead$, the last fully-connected layer for D is denoted as $DHead$, thus, the discriminator D is composed of N and $DHead$, and the Q network is composed of N and $QHead$.

$QHead$ includes some fully-connected layers and outputs the mean and variance of the $\boldsymbol{c}$ distribution. $DHead$ only contains one fully-connected layer activated by Sigmoid and outputs a score between 0 and 1 indicating the extent to which the data is real. Within the shared network N, we separate it into two parts—N_x and N_h, each of which contains separated convolutional layers aligned with batch normalizations and activation functions to deal with the input $\boldsymbol{x}$ or $\boldsymbol{h}$ independently. And here we need guarantee that the architecture of N_h is the same as the architecture of E.

2.3.3.5 Training and Testing Algorithms

In order to get a good embedding network and avoid the problems mentioned previously, a novel training algorithm for C^3-GAN is also designed to guarantee the network stability, where part of the shared convolutional layers are used to update the embedding network periodically aiming to encourage good representation and avoid divergence. In this section, we first explain the training stabilization trick, and then introduce our training and testing algorithm.

Training stabilization trick with E. To solve the stability challenge, we propose to

update E periodically by copying N_h parameters, which means we first fix E and train G, D and Q with our final objective Eq. 2.27, and then copy the parameters of N_h to E every k iterations. In this way, we not only avoid the problems, but also get a good embedding network and stable training performance. Next, we list the reasons why E can be updated using N_h parameters, and then we answer the question why E needs to be updated periodically:

(1) N_h is used to deal with the original condition $\boldsymbol{h}$ independently, thus, the inputs and outputs of N_h and E are in the same form;

(2) N_h and E have the same architecture, the convolutional layers inside N_h apply filters to capture spatial patterns of $\boldsymbol{h}$, the well-updated convolutional layers can also help to find good representations of the original conditions;

(3) since N contains shared layers between D and Q, N_h is in fact a part of Q. Based on the objective of Q in Eq.2.27, N_h is trained to maximize the mutual information between $\boldsymbol{c}$ and $(\boldsymbol{x}, \boldsymbol{h})$ pairs, which partially reduces the information loss between $\boldsymbol{c}$ and $\boldsymbol{h}$ and thus helps to find good representations of $\boldsymbol{h}$. Therefore, it is feasible to update E with N_h, and in this way E is guaranteed to be updated towards a good direction.

Besides, updating E periodically can avoid divergence and improve training stability (as validated in [65]). Since we know keeping updating E online makes the final objective different and lead to divergence, by contrast, fixing E for some iterations and updating it periodically makes the whole network stick to our final objective in Eq. 2.27, and also makes divergence or oscillations much more unlikely. Next, we introduce the details of the training and testing algorithms.

C^3-GAN Training algorithm. Based on our final objective Eq. 2.27, we can get the objective functions for D, G and Q separately. Denote $\boldsymbol{\theta}$ as the parameters of D, η as the

learning rate, we update the discriminator D with Eq. 3.5:

$$\begin{aligned}
&\mathcal{L}_D(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D(\boldsymbol{x}, \boldsymbol{h})] \\
&+ \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z}, \boldsymbol{c}), \boldsymbol{h}))], \\
&\boldsymbol{\theta} = \boldsymbol{\theta} + \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_D(\boldsymbol{\theta}).
\end{aligned} \tag{2.29}$$

For the generator G, we denote $\boldsymbol{\psi}$ as the parameters of G, the loss function and updating rule for G is as follows:

$$\begin{aligned}
&\mathcal{L}_G(\boldsymbol{\psi}) = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z}, \boldsymbol{c}), \boldsymbol{h}))] \\
&- \lambda \mathbb{E}_{\boldsymbol{x}\sim G(\boldsymbol{z},\boldsymbol{c}), \boldsymbol{h}\sim p_{\text{data}}} \left[\mathbb{E}_{\boldsymbol{c}'\sim P(\boldsymbol{c}|\hat{\boldsymbol{x}})} \left[\log Q\left(\boldsymbol{c}' \mid \hat{\boldsymbol{x}}\right)\right]\right], \\
&\boldsymbol{\psi} = \boldsymbol{\psi} - \eta \nabla_{\boldsymbol{\psi}} \mathcal{L}_G(\boldsymbol{\psi}).
\end{aligned} \tag{2.30}$$

For the inference network Q, we denote $\boldsymbol{\omega}$ as the parameters of Q, we update Q with Eq. 4.9:

$$\begin{aligned}
&\mathcal{L}_Q(\boldsymbol{\omega}) = -\lambda \mathbb{E}_{\boldsymbol{x}\sim G(\boldsymbol{z},\boldsymbol{c}), \boldsymbol{y}\sim p_{\text{data}}} \left[\mathbb{E}_{\boldsymbol{c}'\sim P(\boldsymbol{c}|\hat{\boldsymbol{x}})} \left[\log Q\left(\boldsymbol{c}' \mid \hat{\boldsymbol{x}}\right)\right]\right], \\
&\boldsymbol{\omega} = \boldsymbol{\omega} - \eta \nabla_{\boldsymbol{\omega}} \mathcal{L}_Q(\boldsymbol{\omega}).
\end{aligned} \tag{2.31}$$

The detailed training process is shown in Algorithm 1, where D is updated in line 6, G and Q are updated in line 7 and line 8, respectively. For the embedding network E, we update it using N_h parameters every k iterations. More precisely, every k iterations we copy the network N_h to the embedding network E and use the new E to transform original conditions to latent vectors for the following k updates.

C^3-GAN Testing algorithm. When testing the model, we feed the selected new conditions into the embedding network and get the latent vectors, which then go through the

Input: Total training iterations K, E updating period k ($k < K$), a training set, initialized G, D (including N and $DHead$), Q (including N and $QHead$) and E.
Output: Well trained G, D, Q and a good embedding network E.

In each training iteration $iter$:
repeat
  Sample a batch of data $\{(\boldsymbol{x}, \boldsymbol{h})\}$ from the training set.
  Transform $\{\boldsymbol{h}\}$ to a batch of latent vectors $\{\boldsymbol{c}\}$ with E.
  Sample a batch of noise vectors $\{\boldsymbol{z}\}$ from Gaussian distribution.
  Update D with Eq. 3.5.
  Update G with Eq. 3.6.
  Update Q with Eq. 4.9.
  if $iter \bmod k = 0$ **then** Copy N_h parameters to E.
  end if
until $iter > K$.

Algorithm 3: C^3-GAN Training Process

well-trained generator with random noises and get our final estimations (See Alg 4).

Input: New conditions $\{\tilde{\boldsymbol{h}}\}$, well-trained E and G.
Output: Traffic estimations $\{\tilde{\boldsymbol{x}}\}$.

Transform $\{\tilde{\boldsymbol{h}}\}$ to latent vectors $\{\tilde{\boldsymbol{c}}\}$ with E.
Sample noise vectors $\{\boldsymbol{z}\}$ from Gaussian distribution.
Output traffic estimations $\tilde{\boldsymbol{x}} = G(z, \tilde{\boldsymbol{c}})$ with generator G.

Algorithm 4: C^3-GAN Testing Process

2.3.4 Evaluation

In this section, we will first introduce the real-world datasets we use, and then present baselines and evaluation metrics. At last, we provide our experimental results.

2.3.4.1 Dataset and Experiment Descriptions

Dataset Descriptions. We validate our model on real-world datasets: (1) traffic speed, (2) taxi inflow, (3) bus routes and (4) travel demand. The detailed information of the dataset is shown in Table 3.1.

- **Traffic speed.** The hourly average traffic speed is extracted from GPS records collected from taxis in Shenzhen, China from Jul 1st to Dec 31st, 2016. The whole city is partitioned into 40×50 grid cells with a side-length $l_1 = 0.0084°$ in latitude and $l_2 = 0.0126°$ in longitude, and the traffic status in each grid cell is measured by average traffic speed. The data size is (1944, 1, 40, 50), which means there are 1944 traffic distributions in total, and each traffic distribution is a 40×50 matrix.

- **Taxi inflow.** The data is collected from taxis in Shenzhen, China from July 1st to Dec. 31st, 2016. Taxi inflow is the count of all taxis that stayed or arrived at each grid cell within one hour. The data size is (1944, 1, 40, 50), which indicates there are 1944 traffic distributions (of 1-hour slot) in total, each traffic distribution is a 40×50 matrix.

- **Bus routes.** The bus data is collected from 20 different bus routes in Shenzhen, China from July 1st to Dec 31st, 2016. Since there are 990 bus routes in total in Shenzhen City, and only a few of them got updated during July 1st to Dec 31st, 2016, thus, we randomly sample 20 bus routes for simplicity which includes both the unchanged and updated ones. For each bus route, we have the bus route map which is also divided into 40×50 grid cells, and the value of each grid cell indicates the number of buses passing this area within one hour. The data size is (1944, 20, 40, 50), which indicates there are 1944 time slots (1-hour), each time slot has 20 bus route maps, and each bus route map is a 40×50 matrix, so the data dimension for each time slot is $20 \times 40 \times 50$.

- **Travel demand.** The travel demand data is collected from taxis GPS records in Shenzhen, China from July 1st to Dec. 31st, 2016. To extract the travel demands, in each time slot of a day, *i.e.*, one hour, we count the total taxi pickup events within each grid cell. In general, it is hard to obtain the total travel demand including all transport modes. In this work, we use the demand for taxis to represent the local travel demand, where many studies have shown that taxi demands represent the total demands quite

Table 2.7: Dataset descriptions.

Dataset	Timespan	Data size	Dimension
Traffic speed	07/01/2016-12/31/2016	(1944, 1, 40, 50)	40×50
Taxi inflow	07/01/2016-12/31/2016	(1944, 1, 40, 50)	40×50
Bus routes	07/01/2016-12/31/2016	(1944, 20, 40, 50)	$20 \times 40 \times 50$
Travel demand	07/01/2016-12/31/2016	(1944, 1, 40, 50)	40×50

well [2, 14, 15]. The data size is also (1944, 1, 40, 50), which indicates there are 1944 travel demand maps (in 1-hour slot), and each travel demand map is a 40×50 matrix.

Experiment Descriptions. We introduce all different traffic estimation experiments we conducted below.

- **Task 1: traffic speed and taxi inflow estimation based on bus route changes.** In this task, we study how the bus route changes (as urban condition) influence the traffic, thus, we estimate the traffic distributions in Shenzhen City given the complex bus routes as conditions which should be $20 \times 40 \times 50$ tensors, and traffic speed and taxi inflow are both estimated. All the data including traffic speed, taxi inflow and bus route data is divided into training set (90% of data) and testing set (the remaining 10%).

- **Task 2: traffic speed and inflow estimation based on travel demand changes.** In this task, we study how the travel demand changes influence the traffic status, thus, we estimate the traffic distributions in Shenzhen City given the complex travel demands as conditions which are 40×50 matrices. All the data including traffic speed, taxi inflow and travel demand data is also divided into training set (90% of data) and testing set (the remaining 10%).

2.3.4.2 Baselines

To evaluate the effectiveness of our model, we compare our C^3-GAN with state-of-the-art methods[1]. We first use the following two baselines to validate that standard cGAN cannot

[1]Here we do not compare our model with InfoGAN since InfoGAN cannot be used for conditional generation and thus cannot estimate traffic using urban conditions at all.

successfully estimate urban traffic based on complex urban conditions:

- **cGAN [33].** This is the standard conditional GAN, which applies convolutional layers inside both generator and discriminator.
- **cGAN+L1 [66].** This method uses standard conditional GAN structure. The objective of discriminator stays unchanged, while the generator is trained using both the adversarial loss and the L1 loss.

Then, we use two baseline methods to validate that cGAN with a single embedding network E or a Q network is not good enough to solve the traffic estimation problem with complex conditions:

- **cGAN+E [67].** This method uses a predefined embedding network E (namely, a randomly selected network) to transform the original complex conditions to low-dimensional conditions. With the embedded conditions in a latent space, the standard conditional GAN is applied.
- **IcGAN [68, 69].** Invertible conditional GAN (IcGAN) adds an encoder Q to the standard conditional GAN structure, where Q aims to inverse the mapping of a cGAN, *i.e.*, mapping a image into a latent space and a conditional representation.

Next, to evaluate how our C^3-GAN addresses the practical challenge, namely, training stability, we further compare our model with two naive approaches:

- **cGAN+Q+E1 [33, 62]**. This method has the same architecture with C^3-GAN, while E is randomly chosen embedding network. The objective is the same as Eq. 2.27.
- **cGAN+Q+E2 [33, 62].** This method has the same architecture with C^3-GAN, while all the four neural networks including G, D, Q, and E are jointly trained with objective Eq. 2.28.

Besides, we also have state-of-the-art GAN models for traffic estimation as baselines:

- **Curb-GAN [2, 70].** Curb-GAN applies self-attention and convolutional layers to deal with sequential data generation problem, here only convolutional layers are applied to generate average traffic estimations. The generator is trained with the adversarial loss and the L2 loss together.
- **TrafficGAN [1]** TrafficGAN solves the conditional traffic estimation problem, where the conditions are simple discrete values. TrafficGAN applies several dynamic convolutional layers inside generator and discriminator to capture spatial patterns of traffic.

2.3.4.3 Evaluation Metrics

In our experiments, mean absolute percentage error (MAPE) and rooted mean square error (RMSE) are used to evaluate our model:

$$\begin{aligned} \text{MAPE} &= \frac{1}{n_s}\sum_{i=1}^{n_s} |y_i - \hat{y}_i| / y_i, \\ \text{RMSE} &= \sqrt{\frac{1}{n_s}\sum_{i=1}^{n_s} (y_i - \hat{y}_i)^2}, \end{aligned} \tag{2.32}$$

where n_s is the total number of grid cells in the target city area, y_i is the ground-truth traffic status observed in one grid cell s_i, and $\hat{y}_i$ is the corresponding predicted result.

2.3.4.4 Experimental Settings

In the experiment, since we parametrize the auxiliary distribution $Q(\boldsymbol{c}|(G(\boldsymbol{z},\boldsymbol{c}),\boldsymbol{h}))$ as a neural network, its form depends on the true posterior $P(\boldsymbol{c}|(G(\boldsymbol{z},\boldsymbol{c}),\boldsymbol{h}))$. We found that simply treating $Q(\boldsymbol{c}|(G(\boldsymbol{z},\boldsymbol{c}),\boldsymbol{h}))$ as a factored Gaussian distribution is sufficient.

Table 2.8: Performance on task 1: traffic speed and taxi inflow estimation based on bus route changes.

Methods		cGAN	cGAN+L1	Curb-GAN	TrafficGAN	cGAN+E	IcGAN	cGAN+Q+E1	cGAN+Q+E2	C^3-GAN
Traffic speed	RMSE	49.08	36.81	44.52	108.78	104.41	80.94	29.05	62.42	**16.79**
	MAPE	6.68	2.92	3.32	62.43	60.76	42.38	2.48	45.75	**1.64**
Taxi inflow	RMSE	524.62	415.78	730.66	1579.56	1494.72	1771.25	251.23	293.08	**143.07**
	MAPE	65.67	15.38	64.66	670.33	649.39	631.32	14.58	36.31	**10.86**

Table 2.9: Performance on task 2: traffic speed and taxi inflow estimation based on travel demand changes.

Methods		cGAN	cGAN+L1	Curb-GAN	TrafficGAN	cGAN+E	IcGAN	cGAN+Q+E1	cGAN+Q+E2	C^3-GAN
Traffic speed	RMSE	19.73	60.07	48.90	69.31	23.37	84.17	18.32	70.02	**13.75**
	MAPE	3.24	12.42	8.34	39.39	6.62	45.40	2.83	45.23	**1.31**
Taxi inflow	RMSE	170.09	300.15	758.76	1364.87	1570.29	1872.08	28.05	410.71	**19.81**
	MAPE	10.98	30.96	165.10	430.21	381.00	799.81	5.75	97.32	**4.54**

For all experiments, we use Adam[56] for online optimization and apply batch normalization[71] after most layers. The detailed structure of C^3-GAN in our experiments is as follows: the generator G contains 4 convolutional layers with kernel sizes $\{5, 7, 7, 6\}$ and output channels $\{1024, 128, 64, 1\}$; the discriminator D and the inference network Q share some layers denoted as N which is partitioned into N_x and N_h, N_x includes four convolutional layers with kernel sizes $\{6, 7, 7, 5\}$ and output channels $\{64, 128, 1024, 128\}$; N_h includes four convolutional layers with kernel sizes $\{6, 7, 7, 5\}$ and output channels $\{64, 128, 1024, 128\}$ and one fully-connected layer; $DHead$ contains only one fully-connected layer and $QHead$ contains 3 fully-connected layers, besides, the embedding network E has the same structure as N_h.

2.3.4.5 Results

Overall performance results. In this part, we present the overall performance of our C^3-GAN compared with all baseline models on two different traffic estimation tasks. For all deep models, we train and test all methods five times, and we pick the best trained model and show the testing results including RMSE and MAPE. We have the following observations based on all statistics from both tasks.

First, the overall performance of task 1 is presented in Table 3.2, which includes both

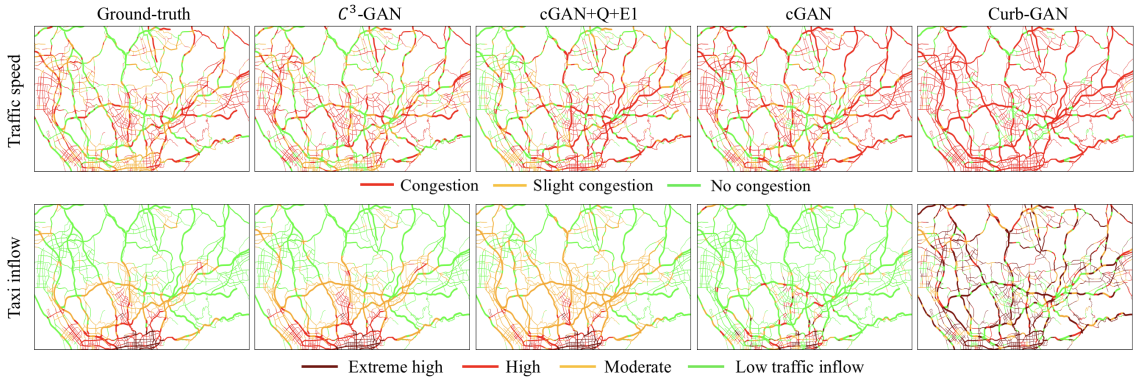


Figure 2.29: Visualizations of (1st row) traffic speed with bus route condition & (2nd row) taxi inflow with travel demand condition.

traffic speed and taxi inflow estimation based on bus route changes, we find the classic baseline models including cGAN, cGAN+L1, Curb-GAN and TrafficGAN present high testing errors (*i.e.*, high RMSE and high MAPE), which indicates these methods cannot deal with the conditional generation regarding to complex conditions very well. Besides, compared with our C^3-GAN, the two baseline models incluing cGAN+E and IcGAN still present bad performance, which means it is not enough to only equip cGAN with a simple randomly pre-defined embedding network or an inference network. Furthermore, the high testing error of the baseline cGAN+Q+E1 tell us it is very important to find a proper embedding network. As for the baseline model cGAN+Q+E2, the reason why it produce high testing errors is the embedding network keeps updating online, which easily leads to divergence and instability during training process. The performance of task 2 is shown in Table 3.3 which presents similar results.

Traffic estimation visualizations. To further validate the effectiveness of our C^3-GAN, we visualize the traffic distributions over the road networks. In both tasks, the well-trained C^3-GAN and baseline models are tested on our test set, given a certain urban condition, we compare the average generated results of each model with the ground truth.

As shown in Figure 3.4, when estimating the traffic speed regarding to bus route changes, we visualize the ground-truth traffic distributions in Shenzhen City and the cor-

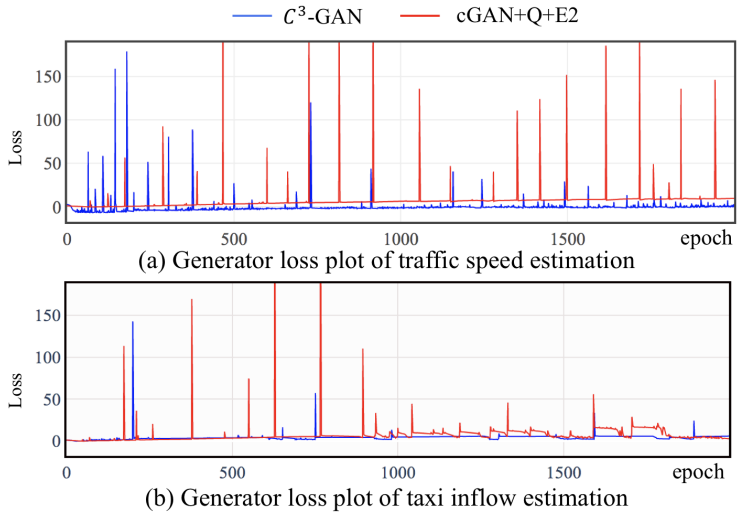


Figure 2.30: Loss plots of traffic speed and taxi inflow estimation regarding to bus route changes.

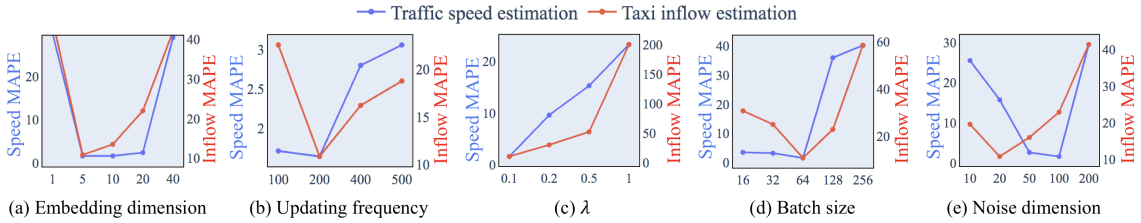


Figure 2.31: Impact of parameters on traffic speed and taxi inflow estimation with bus route conditions.

responding estimation results generated by our C^3-GAN and some competitive baselines including cGAN, Curb-GAN and cGAN+Q+E1. Clearly, cGAN and Curb-GAN output low quality traffic estimations, since it is hard for both of them to deal with complex conditions. The bad performance of cGAN+Q+E1 indicates that a random embedding network doesn't guarantee good generation results. In contrast, our C^3-GAN generates reasonable traffic distributions compared to the ground-truth, which indicates our C^3-GAN can build a strong connection between the complex conditions and the traffic and thus produce more reliable traffic estimations.

When estimating the taxi inflow regarding to travel demand changes, the visualizations are presented in the second row of Figure 3.4. It is obvious that our C^3-GAN pro-

duces the most realistic traffic estimation compared to the ground truth, and outperforms other competitive baselines. For brevity, we omit the visualizations of taxi inflow estimation based on bus route changes and the traffic speed estimation based on travel demand changes, both of which have similar results to Figure 3.4.

Model stability. From previous experimental results, we have shown our C^3-GAN is effective in traffic estimation, next, we will study the stability of our model. In Section 3.4, we explained the reason why our C^3-GAN can improve the model stability compared with the naive approach which has trainable embedding net mentioned in Section 3.3. Our results below provide experimental evidence to prove that C^3-GAN can improve the training stability and avoid divergence compared with the baseline cGAN+Q+E2 which is exactly the naive approach mentioned in Section 3.3.

As shown in Figure 2.30, we present two typical loss plots of our C^3-GAN and the baseline cGAN+Q+E2, both of the models are trained for 2000 epochs. Figure 2.30(a) and Figure 2.30(b) are the generator training loss of our C^3-GAN and cGAN+Q+E2 in the experiments of traffic speed and taxi inflow estimation regarding to bus route changes. Apparently, cGAN+Q+E2 is easy to diverge (Figure 2.30(a)) or be unstable (Figure 2.30(b)), since the losses usually present great vibrations after 1500 epochs. In contrast, our C^3-GAN converges much faster, which also demonstrates that the architecture and training algorithm of C^3-GAN are successful in improving model stability.

Evaluations on hyper-parameters. Since our C^3-GAN contains many different hyper-parameters including the dimension of embedded vectors, updating frequency of the embedding network, batch size, *etc*. Next, we evaluate the impacts of different hyper-parameters and present the evaluation results. For all hyper-parameters, each time we only adjust one of them with the others fixed.

As shown in Figure 4.7(a), we fixed all other parameters of C^3-GAN and adjust the dimension of embedded vectors, we find that the estimation performance is sensitive to

the embedding dimension. For both of speed and inflow estimation regarding to bus route changes. Obviously, low embedding dimension presents bad performance, since the low-dimensional embedded vectors are not good representations of the original conditions. Besides, the high embedding dimensions usually result in worse performance, which indicates it is harder to build strong connections between more complex latent vectors and the traffic.

Figure 4.7(b) shows the estimation performance with different updating frequency of the embedding network, and we get high errors when we update E with very high or low frequency. The high updating frequency of E will lead to the situation where the whole model keeps training without getting a better embedding network, the low updating frequency will influence the training efficiency of the whole network, both of the scenarios present bad performance or longer training time.

Figure 4.7(c) is the estimation performance based on different λ. λ should be chosen based on the cGAN objective scale, improper λ would affect the final generation performance, and in our experiments, the best choice of λ is 0.1.

As shown in Figure 4.7(d), we find that the estimation performance is sensitive to batch size. Large batch size results in bad estimation performance in our experiments, which also matches the conclusions in the work [72] stating when using a large batch, there is a significant degradation in the generalization ability of the model.

Last but not the least, we evaluate the impact of the random noise dimension. As shown in Figure 4.7(e), we find the estimation errors are high when the noise dimension is very low or high. With low noise dimension, the fewer weights within the network are not enough to learn the spatial dependencies of traffic. In contrast, if the noise dimension is too high, more weights need to be trained and more training epochs are needed.

3

Transferable Generative Adversarial Networks

3.1 Spatially-Transferable Generative Adversarial Networks

3.1.1 Overview

3.1.1.1 Introduction

Conditional traffic estimation is a critical problem in urban development, especially in land use planning, subway routes planning, *etc*. Given the road network of a city, its historical traffic status, and an urban development plan which usually leads to new local travel demands, the *conditional urban traffic estimation problem* aims to accurately estimate the future traffic status based on the changing travel demands. Solving such problem not only provides insights to evaluate the feasibility of the urban deployment plan, but also helps to reduce potential traffic congestion and improves local transportation efficiency.

A number of data-driven methods have achieved success on single-city conditional

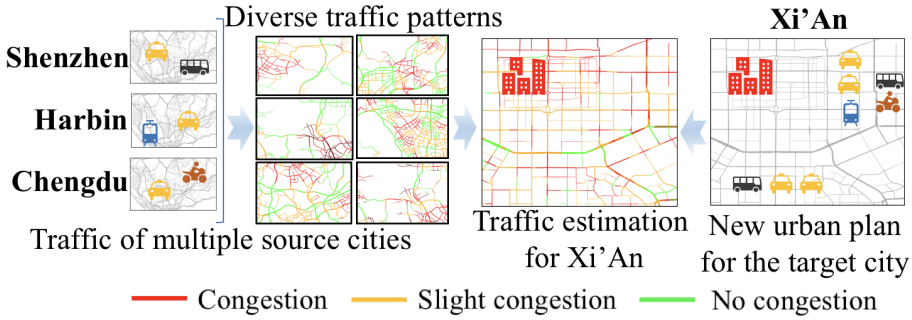


Figure 3.1: An example of conditional traffic estimation by transferring knowledge from multiple source cities (Shenzhen, Harbin, etc) to the target city (Xi'An).

traffic estimation, such as classical machine learning models [7, 8, 57] or GAN-based models including CurbGAN [2] and TrafficGAN [1]. However, these methods typically require large amount of training data of the target city and are unable to perform well in case of data scarcity, *e.g.*, when estimating the traffic in a previously unseen city or a newly-built region. A simple idea to address the data scarcity issue would be to "borrow" enough training data from other cities if they are available. However, due to spatial heterogeneity and local traffic pattern differences, such tricks usually result in poor estimation quality.

An effective solution to the data scarcity problem is to employ a transfer learning paradigm, which avoids training models directly using the limited data of a "new" city but transfers urban knowledge from other source cities (with abundant data) to the target city (with insufficient data) to enable good performance [73]. *In this paper, we aim at developing a spatial transfer generative learning framework for cross-city conditional urban traffic estimation in case of data scarcity.*

Prior art. Unfortunately, existing spatial transfer learning techniques may not directly solve our cross-city conditional traffic estimation problem. For example, RegionTrans [74], TL-DCRNN [75] and STCNet [76] borrow the transfer leaning framework to forecast future traffic using historical time-series traffic data. However, in most of the urban transfer leaning methods, only one source city is used to extract transferred knowledge. To enable

good transfer performance, these methods have to guarantee that the source city and the target city have a lot in common, or it would turn out to be a failure if the two cities are too different [77].

Some works try to transfer knowledge from multiple cities, which potentially increases the diversity of the source data and thus avoids the high-similarity constraint between source cities and target cities. For example, a new transfer framework [78] tries to generate mobility data for a target city by transferring knowledge from multiple source cities. MetaST [79] focuses on time series traffic prediction using multiple cities as source cities. However, both of them cannot solve the conditional traffic estimation problem, since they did not consider the impact of travel demands and the diverse traffic patterns in different cities.

Our insight. When solving the cross-city conditional traffic estimation problem in the transfer learning paradigm, compared to one single source city, the shared knowledge extracted from multiple source cities would contain more comprehensive traffic patterns, and provide more information about how traffic status changes along the complex road networks in different regions. Therefore, transferring from multiple source cities may greatly improve the estimation accuracy and transfer stability. For example, as illustrated in Figure 4.1, with a new urban construction plan to be evaluated in Xi'An City which only has little traffic data available, we can borrow the traffic knowledge extracted from multiple source cities (*e.g.*, Shenzhen, Harbin and Chengdu) to provide more accurate traffic estimations for Xi'An.

Challenges. In this paper, we study the problem of conditional traffic estimation in case of data scarcity by transferring knowledge from multiple source cities. However, such cross-city conditional traffic estimation problem is hard to solve due to two key challenges:

(1) *Knowledge extraction and transfer.* The traffic patterns between travel demands and

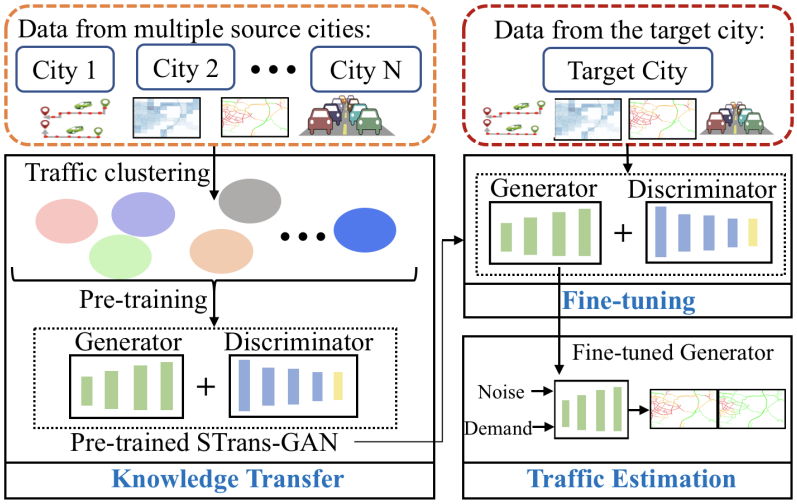


Figure 3.2: Solution framework.

local traffic status highly depend on the complex road networks and vary from region to region and time to time, which lead to the difficulties in knowledge extraction and transfer. When multiple cities are considered as source cities, the traffic patterns become even more complicated and thus harder to learn and transfer.

(2) *Knowledge adaptation.* The extracted knowledge should be well-adapted to the target city. Since different regions of the target city in different time slots could have different traffic patterns, the extracted knowledge should be adapted to all these scenarios in different ways. Thus, how to perform the knowledge adaption in a flexible manner is very important but challenging.

Contributions. To solve the cross-city conditional urban traffic estimation problem in case of data scarcity, we propose to estimate traffic in a data generation perspective under the multiple-city transfer learning setup. Hence, a new spatial transfer generative learning framework — Spatially-Transferable Generative Adversarial Networks (STrans-GAN) is proposed, which successfully tackles both of the aforementioned challenges and provides accurate traffic estimations based on various travel demands. Figure 4.2 is our solution framework. To solve the first challenge, traffic clustering is performed to aggregate all

historical traffic data from multiple source cities into different clusters based on their traffic patterns, and then in the pre-training process, meta-learning idea is incorporated to learn a good global-initialized model from all clusters. To address the second challenge, the pre-trained STrans-GAN can be adapted to any scenario of the target city with only few samples by adding an extra cluster matching regularizer. Our main contributions can be summarized as follows:

- To the best of our knowledge, we are the first to solve the cross-city conditional traffic estimation problem in case of data scarcity from a spatial transfer generative learning perspective, and propose a novel method — Spatially-Transferable Generative Adversarial Networks (STrans-GAN).
- STrans-GAN preserves various traffic patterns from multiple source cities through traffic clustering, and incorporates meta-learning idea into the pre-training process to learn a global generalized model, which targets the first challenge. During fine-tuning, a new cluster matching regularizer is added to realize the flexible adaptation in different scenarios and thus addresses the second challenge.
- Extensive experiments on multiple-city datasets are performed to evaluate the effectiveness of our STrans-GAN. The experimental results prove that STrans-GAN significantly improves the urban traffic estimation performance and outperforms state-of-the-art baselines.

3.1.1.2 Preliminaries

In this section, we first introduce the definitions and then formally define our problem.

Definition 1 (Grid cells). A city is partitioned into $m_1 \times m_2$ grid cells, each grid cell has equal side-length in latitude and longitude. The set of grid cells of one city is defined as $\mathbb{S} = \{s_{ij}\}$, where $\langle i, j \rangle$ indicates the coordinates of the grid cell s_{ij}, $1 \leq i \leq m_1$ and

$1 \leq j \leq m_2$.

Definition 2 (Target region). A target region R is a square geographic region in a city, formed with $r \times r$ grid cells. A whole city can be split into multiple regions $\mathcal{R} = \{R_{ij}\}$, where $\langle i, j \rangle$ indicates the coordinates of the top-left grid cell in the region R_{ij}.

Definition 3 (Travel Demand). The travel demand of an area captures the total number of departures in a period of time. Thus, during one time period, the travel demand of a grid cell s is denoted as $d_s \in \mathbb{N}_0$, and the travel demand of a target region is denoted with a matrix $\boldsymbol{d}_R \in \mathbb{N}_0^{r \times r}$, where each entry d_s within the matrix $\boldsymbol{d}_R$ indicates the corresponding travel demand of the grid cell s within the region R. In this study, we use the demand for taxis to represent travel demand just as many literature works [1, 2, 15].

Definition 4 (Traffic status and traffic distribution). Traffic status is the basic knowledge of the road network traffic at a grid cell, which can be measured by traffic speed, traffic inflow/outflow, *etc*. We denote x_s as the average traffic status of a grid cell s within a period of time, and denote $\boldsymbol{x}_R \in \mathbb{R}^{r \times r}$ as the traffic distribution matrix of the region R, which is an $r \times r$ matrix composed of traffic status of all grid cells within the region.

Problem Statement: Given multiple source cities $\mathcal{U}_{\text{source}} = \{u^i_{\text{source}}\}$ and one target city u_{target} partitioned into grid cells, historical samples of travel demands $\mathcal{D}_{\text{source}} = \{\boldsymbol{d}_R\}$ and traffic distributions $\mathcal{X}_{\text{source}} = \{\boldsymbol{x}_R\}$ from all source cities, and only a small amount of historical samples of travel demands $\mathcal{D}_{\text{target}} = \{\boldsymbol{d}_R\}$ and traffic distributions $\mathcal{X}_{\text{target}} = \{\boldsymbol{x}_R\}$ from the target city, we aim to estimate the future traffic distributions $\{\tilde{\boldsymbol{x}}_R\}$ for a set of new travel demands $\{\tilde{\boldsymbol{d}}_R\}$ from the target city u_{target}.

3.1.2 Related Work

In this section, we summarize the literature works from two related areas: 1) urban traffic estimation, and 2) urban transfer learning.

Urban traffic estimation. In recent years, more and more studies have focused on ur-

ban traffic estimation problem. Some works [7, 8, 17, 18, 19, 57] tried to apply classic machine learning methods to solve this problem. For example, the work [17] proposes to predict traffic volume by combining machine learning techniques and well-established traffic flow theory, and the works [19, 43, 44] propose novel frameworks to predict crowd flows and individual's movement.

Other works borrowed deep learning frameworks to solve various spatial-temporal prediction problems. For example, some works [45, 46] focus on citywide flow prediction and traffic demand prediction using CNN to better capture the spatial dependencies of urban data. Other works such as [41, 48, 49] try to solve travel time prediction and traffic speed prediction problems using recurrent neural networks [47] and LSTM [58] aiming to better capture the temporal dependencies within the data. Moreover, the work [40] tries to predict crowd density with ConvLSTM [51] to capture both spatial and temporal dependencies simultaneously. In addition, many previous works propose to solve the traffic estimation problem using generative adversarial networks [29]. For example, TrafficGAN [1], Curb-GAN [2] are proposed to estimate future traffic in an geographical region, C^3-GAN [3] tries to estimate traffic based on complex traffic related features. However, all these works would fail once lacking training samples.

Urban transfer learning. Transfer learning is a subarea in machine learning, which focuses on extracting and learning knowledge in one problem and applying it to a different but related problem. Transfer learning has been applied to many different urban scenarios, *e.g.*, many urban computing applications including traffic prediction, events detection, and urban deployment borrow transfer learning framework to solve urban data scarcity problem [73].

In previous studies, some works including [74, 75, 76, 77] propose novel traffic prediction frameworks on top of transfer leaning. For example, Wang et al. [80] propose to use transfer learning to solve ride-sourcing car detection problem. Among all these

works, only one source domain is used to extract and learn knowledge. Some other works try to transfer knowledge from multiple source domains to target domains. For example, He et al. propose a novel mobility prediction framework [78] which transfers knowledge learned from multiple source cities to target cities aiming to generate mobility data for the target. Yao et al. [79] try to solve the traffic prediction problem using multiple cities as source cities. However, all these works cannot be generalized to solving conditional traffic estimation problem, and they did not consider the impact of travel demands and diverse traffic patterns.

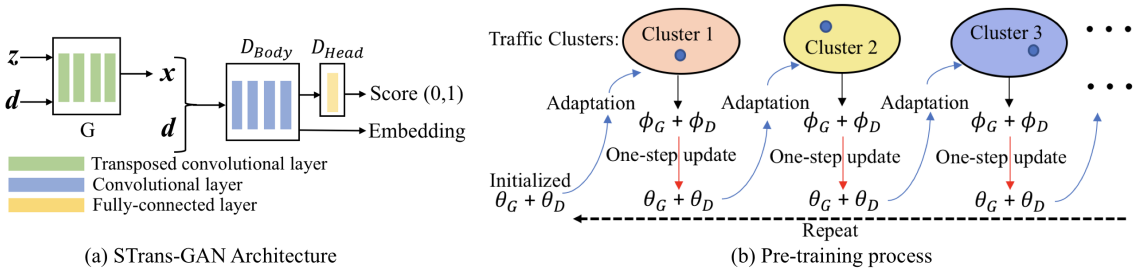
3.1.3 Methodology

Inspired by generative adversarial networks (GAN) [29], we are trying to solve the cross-city conditional traffic estimation problem in a traffic data generation perspective. The state-of-the-art conditional GAN (cGAN) model [33] seems to be a promising method, and we can view travel demands as conditions and traffic distributions as "images". However, training a good cGAN model requires a large amount of training data to ensure the convergence. If we perform traffic estimation in a city which faces data scarcity, cGAN would definitely fail due to the lack of training samples.

Thus, to better solve the cross-city conditional traffic estimation problem in case of data scarcity, we propose a spatial transfer generative learning framework — STrans-GAN which combines the generative model with multiple-city transfer learning paradigm. STrans-GAN also addresses the aforementioned challenges with novel designs:

(1) *Knowledge extraction and transfer.* To tackle the first challenge, we propose a unique architecture and a pre-training algorithm. STrans-GAN preserves various traffic patterns through traffic clustering, and incorporates meta-learning idea into the pre-training process to learn a well-initialized model.

(2) *Knowledge adaptation.* During fine-tuning, we propose to add an extra cluster match-



(a) STrans-GAN Architecture

(b) Pre-training process

Figure 3.3: STrans-GAN Overview.

ing regularizer to realize the flexible adaptation in different scenarios of the target city. Besides, a novel fine-tuning algorithm is proposed.

In this section, we first introduce our STrans-GAN architecture, and then detail the novel knowledge transfer and fine-tuning processes.

3.1.3.1 Model Architecture

STrans-GAN is a framework designed to solve the cross-city conditional urban traffic estimation problem in the multiple-city transfer learning setup. Since the effectiveness of GANs in traffic estimation has been proved in recent studies [1, 2], in this work, we design our STrans-GAN on top of cGAN. STrans-GAN has convolutional layers inside to better capture the complex spatial traffic dependencies, and combines the adversarial loss with L1 regularizer to improve the estimation performance during pre-training process. Figure 3.3(a) is the detailed architecture of our STrans-GAN, which is composed of a generator G and a discriminator D.

The generator G aims to learn a distribution $\boldsymbol{x} \sim G(\boldsymbol{z}, \boldsymbol{d})$ which matches the real data distribution p_{data}. The input of the generator G includes i) a noise vector $\boldsymbol{z}$ randomly sampled from Gaussian distribution, *i.e.*, $\boldsymbol{z} \sim p_{\boldsymbol{z}}$, and ii) the travel demand $\boldsymbol{d}$. G outputs the generated traffic distribution $\boldsymbol{x} \sim G(\boldsymbol{z}, \boldsymbol{d})$. Inside the generator, $\boldsymbol{d}$ and $\boldsymbol{z}$ are first concatenated together and then pass the stacked transposed convolutional layers, all the layers are batch normalized and activated by ReLU or hyperbolic tangent function.

The discriminator D aims to distinguish the real traffic distributions from the generated ones by giving a high score if the input $\boldsymbol{x}$ is sampled from the training set and matches the travel demand $\boldsymbol{d}$, or producing a low score if the input $\boldsymbol{x}$ is a generated one or does not match the input $\boldsymbol{d}$. The discriminator has two components including D_{Body} and D_{Head}, the input of D_{Body} includes i) a traffic distribution $\boldsymbol{x}$, which could be sampled from the dataset or generated by the generator, *i.e.*, $\boldsymbol{x} \sim p_{\text{data}}$ or $\boldsymbol{x} \sim G(\boldsymbol{z}, \boldsymbol{d})$, and ii) the travel demand $\boldsymbol{d}$. Inside D_{Body}, $\boldsymbol{d}$ and $\boldsymbol{x}$ are concatenated together and pass all stacked convolutional layers, all the layers are batch normalized and activated by leaky ReLU or hyperbolic tangent function. The output of D_{Body} is a vector which can be viewed as the embedding of the input pair (*i.e.*, travel demand and traffic distribution), the embedding vector passes D_{Head} which contains a single fully-connected layer to get the final score.

The basic objective function is as Eq. 3.1:

$$\begin{aligned}\mathcal{L}_{cGAN}(G, D) =& \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D(\boldsymbol{x}, \boldsymbol{d})] \\ &+ \mathbb{E}_{\boldsymbol{z}\sim p_z}[\log(1 - D(G(\boldsymbol{z}, \boldsymbol{d}), \boldsymbol{d}))].\end{aligned} \tag{3.1}$$

To avoid overfitting and improve the estimation performance, we combine the Eq. 3.1 with L1 regularizer [66]:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x\sim p_{\text{data}}, \boldsymbol{z}\sim p_z}\left[\|\boldsymbol{x} - G(\boldsymbol{z}, \boldsymbol{d})\|_1\right], \tag{3.2}$$

thus, the final objective is as Eq. 3.3:

$$\mathcal{L}(G, D) = \mathcal{L}_{cGAN}(G, D) + \alpha\mathcal{L}_{L1}(G), \tag{3.3}$$

and the optimization process is a min-max game, *i.e.*, $\min_G \max_D \mathcal{L}(G, D)$, where the generator tries to minimize the loss while the discriminator tries to maximize it.

3.1.3.2 Knowledge Transfer

The STrans-GAN introduced above only works when enough training data available, once data scarcity appears, it is hard to get a well-trained model. To better solve the traffic estimation problem especially in a city facing data scarcity problem, we propose to enable the STrans-GAN to learn from multiple source cities, which means the knowledge learned from source cities can be subsequently transferred to new target cities.

However, knowledge extraction and transfer from multiple source cities to the target city is very challenging. Given the traffic data (including traffic status and travel demands) and the road networks from multiple source cities, the key steps of knowledge transfer includes (i) disentangling and preserving different traffic patterns from multiple sources, and (ii) learning an initialized STrans-GAN by which the learned knowledge can be transferred. Thus, we first perform traffic clustering to detect different traffic patterns, and then design a novel pre-training algorithm which incorporates meta-learning idea to learn a well-initialized STrans-GAN.

Traffic Clustering. Each region in each source city may present various traffic patterns in different time slots, if the traffic distributions from all source cities are simply mixed together, the diversities of traffic patterns are ignored, which usually leads to very limited improvements than leaning from scratch (proved in the experiments). Thus, it is important to recognize different traffic patterns from multiple source cities.

Clustering is an effective way to disentangle patterns from data. However, given different source cities, we usually do not know what the proper number of clusters is, and whether the traffic data is clustered in a good way. To study how the number of clusters affect the performance and what number of clusters would be better for the given source cities, we apply k-means [81] to cluster the traffic data (*i.e.*, travel demand and traffic distribution pairs from all source cities). K-means is a flexible clustering method, it would produce different clusters based on different initial centroids, which helps to ex-

plore the best clustering way for different source cities, more empirical proofs are shown in Section 3.1.4.6.

For a region R in a specific time slot, we have a travel demand $\boldsymbol{d}$ and a traffic distribution $\boldsymbol{x}$, and the average travel demand $\bar{d}$ and average traffic status $\bar{x}$ can be viewed as features to perform k-means clustering, where $\bar{d} = (\sum \boldsymbol{d})/r^2$, $\bar{x} = (\sum \boldsymbol{x})/r^2$.

In the traffic clustering, we execute the following four steps alternatively: (1) choose the number of clusters and randomly initialize the centroid for each traffic cluster; (2) assign all the traffic data pairs to their closest cluster centroid; (3) calculate the average feature of all traffic data within the cluster, and assign the new centroid for each newly formed cluster to the data point which is the closest to the average feature, (4) repeat previous two steps. Besides, the number of clusters is a hyper-parameter which will be tuned in the experiments.

Input: Total training iterations K_1, n traffic clusters (*i.e.*, tasks), the innerloop k, initialized $\boldsymbol{\theta}_G$, $\boldsymbol{\theta}_D$, $\boldsymbol{\phi}_G$, and $\boldsymbol{\phi}_D$.
Output: Pre-trained $\boldsymbol{\theta}_G$, $\boldsymbol{\theta}_D$.

for iteration $\leftarrow 1$ to K_1 **do**
  for cluster $\leftarrow 1$ to n **do**
    $\boldsymbol{\phi}_G \leftarrow \boldsymbol{\theta}_G$.
    $\boldsymbol{\phi}_D \leftarrow \boldsymbol{\theta}_D$.
    Sample a batch of data $\{(\boldsymbol{x}, \boldsymbol{d})\}$.
    for innerloop $\leftarrow 1$ to k **do**
      Sample a batch of noise vectors $\{\boldsymbol{z}\}$.
      Update $\boldsymbol{\phi}_D$ with Eq. 3.5.
      Update $\boldsymbol{\phi}_G$ with Eq. 3.6.
    end for
    Update $\boldsymbol{\theta}_D$ one step with Eq. 3.7.
    Update $\boldsymbol{\theta}_G$ one step with Eq. 3.8.
  end for
end for

Algorithm 5: STrans-GAN Pre-training Process

Pre-training. After clustering, the traffic data with similar patterns is clustered together, and the traffic patterns detection is finished, the next step is to find a way to make

the traffic knowledge preserved in each cluster transferable. Thus, we propose a novel pre-training algorithm for our STrans-GAN on top of meta-leaning method Reptile [82], this new algorithm trains a global initialized STrans-GAN using the data from all clusters, and the traffic knowledge is preserved in the parameters of the pre-trained STrans-GAN.

Since the data in each cluster has similar patterns, we can treat each cluster as a traffic estimation task τ and get the estimation loss L_τ using the traffic data (*i.e.*, traffic demand and traffic distribution pairs) within cluster τ. During the pre-training process, our goal is to find the well-initialized generator and discriminator parameters $\boldsymbol{\theta}_G$ and $\boldsymbol{\theta}_D$ using all training tasks (*i.e.*, clusters), which can quickly converge on a new task τ' with little data and few adaptation steps by minimizing the loss $L_{\tau'}$.

The objective of the pre-training process is as follows:

$$\min_{\boldsymbol{\theta}_G, \boldsymbol{\theta}_D} \mathbb{E}_\tau \left[L_\tau \left(U_\tau^k(\boldsymbol{\theta}_G, \boldsymbol{\theta}_D) \right) \right], \tag{3.4}$$

where U corresponds to one step of stochastic gradient descent [83] on D and G with respect to the loss in Eq 3.3, and we denote $\boldsymbol{\phi}_G, \boldsymbol{\phi}_D = U_\tau^k(\boldsymbol{\theta}_G, \boldsymbol{\theta}_D)$ as the adapted parameters of G and D after k steps of gradient descent in task τ.

The detailed pre-training algorithm is as Alg 9. The adapted parameters $\boldsymbol{\phi}_D$ and $\boldsymbol{\phi}_G$ of STrans-GAN for each cluster is updated using Eq 3.5 and Eq 3.6. After adapting $\boldsymbol{\phi}_G$ and $\boldsymbol{\phi}_D$ k steps in the innerloop, we update one step of $\boldsymbol{\theta}_D$ and $\boldsymbol{\theta}_G$ using Eq 3.7 and Eq 3.8.

$$\begin{aligned} \mathcal{L}(\boldsymbol{\phi}_D) &= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [\log D(\boldsymbol{x}, \boldsymbol{d})] \\ &+ \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}} [\log(1 - D(G(\boldsymbol{z}, \boldsymbol{d}), \boldsymbol{d})], \\ \boldsymbol{\phi}_D &= \boldsymbol{\phi}_D + \eta \nabla_{\boldsymbol{\phi}_D} \mathcal{L}(\boldsymbol{\phi}_D). \end{aligned} \tag{3.5}$$

Input: Total fine-tuning iterations K_2, n traffic clusters (*i.e.*, tasks), dataset for fine-tuning, pre-trained $\boldsymbol{\theta}_G$, $\boldsymbol{\theta}_D$.
Output: Fine-tuned $\boldsymbol{\theta}_G$, $\boldsymbol{\theta}_D$.
1: Get the centroid embeddings for each cluster $\boldsymbol{v}_c = D_{\text{body}}(\boldsymbol{x}_c, \boldsymbol{d}_c)$.
2: **for** iteration $\leftarrow 1$ to K_2 **do**
3: Sample a batch of data $\{(\boldsymbol{x}, \boldsymbol{d})\}$ from fine-tuning set
4: Sample a batch of noise vectors $\{\boldsymbol{z}\}$.
5: Update $\boldsymbol{\theta}_D$ with Eq. 3.12.
6: Update $\boldsymbol{\theta}_G$ with Eq. 3.13.
7: **end for**

Algorithm 6: STrans-GAN Fine-tuning Process

$$\mathcal{L}(\boldsymbol{\phi}_G) = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z}, \boldsymbol{d}), \boldsymbol{d}))] + \alpha\mathcal{L}_{L1}(G),$$
$$\boldsymbol{\phi}_G = \boldsymbol{\phi}_G - \eta\nabla_{\boldsymbol{\phi}_G}\mathcal{L}(\boldsymbol{\phi}_G). \tag{3.6}$$

$$\boldsymbol{\theta}_D = \boldsymbol{\theta}_D + \lambda(\boldsymbol{\theta}_D - \boldsymbol{\phi}_D). \tag{3.7}$$

$$\boldsymbol{\theta}_G = \boldsymbol{\theta}_G + \lambda(\boldsymbol{\theta}_G - \boldsymbol{\phi}_G). \tag{3.8}$$

We can get a good global initialized STrans-GAN through this pre-training process since its final outputs $\boldsymbol{\theta}_G$ and $\boldsymbol{\theta}_D$ can reach the point which has minimum distance to each task just as illustrated in Eq 3.9, and this has been proved by many works [82, 84].

$$\min_{\boldsymbol{\theta}_G, \boldsymbol{\theta}_D} \sum_{\tau} |\boldsymbol{\theta}_D - \boldsymbol{\phi}_D^{\tau}| + |\boldsymbol{\theta}_G - \boldsymbol{\phi}_G^{\tau}|. \tag{3.9}$$

In Eq 3.9, $\boldsymbol{\phi}_G^{\tau}$ and $\boldsymbol{\phi}_D^{\tau}$ are the optimal parameters of discriminator and generator for task τ. Hence, if we have a new task τ' which is similar to one of the training tasks, $\boldsymbol{\theta}_G$ and $\boldsymbol{\theta}_D$ can adapt to $\boldsymbol{\phi}_G^{\tau'}$, $\boldsymbol{\phi}_D^{\tau'}$ very fast, and thus the rapid and easy model generalization is realized.

3.1.3.3 Model Fine-tuning

Once we get a well-initialized STrans-GAN, we need to adapt it to the target city. However, a target city would have many different regions, each region would present different traffic patterns during different time slots, and it is common that the traffic distributions for a specific region belong to different traffic clusters. Thus, fine-tuning the pre-trained STrans-GAN with Eq 3.3 is not good enough, since it cannot guarantee that the pre-trained STrans-GAN can be correctly adapted to the corresponding traffic clusters for different traffic distributions.

To fine-tune the STrans-GAN parameters on the target city with few fine-tuning samples, we need to ensure the STrans-GAN can detect the traffic patterns for each sample (*i.e.*, travel demand and traffic distribution pair) and also generate reasonable traffic distributions, in other words, we need to make the fine-tuning process flexible for different traffic patterns. To realize this goal, firstly, we should guarantee the STrans-GAN can produce like-real traffic distributions based on different travel demands, which has been realized by adversarial loss and the L1 loss shown in Eq 3.3. Besides, we add a cluster matching regularizer, which enables the pre-trained STrans-GAN to be automatically adapted to different clusters based on different traffic patterns presented by the data. During fine-tuning, for each data sample of the target city, the cluster matching regularizer tries to minimize the distance between the data sample embedding and its corresponding cluster embedding, which helps the STrans-GAN to produce better generation results and present more clear traffic patterns. The cluster matching regularizer is defined as follows:

$$\begin{aligned} &\mathcal{L}_{\mathrm{c}}(D) = \mathbb{E}\left[\|\boldsymbol{v}_c - \boldsymbol{v}\|_1\right], \\ &\boldsymbol{v}_c = D_{\text{body}}(\boldsymbol{x}_c, \boldsymbol{d}_c), \boldsymbol{v} = D'_{\text{body}}(\boldsymbol{x}, \boldsymbol{d}). \end{aligned} \tag{3.10}$$

In Eq 3.10, for each data sample $(\boldsymbol{x}, \boldsymbol{d})$ from the target city, we first figure out the exact cluster c that the data sample $(\boldsymbol{x}, \boldsymbol{d})$ belongs to by calculating the distance between the data

sample and each cluster centroid and choosing the closest cluster, and then we get the centroid embedding $\boldsymbol{v}_c$ for cluster c by passing the centroid to D_{body}, *i.e.*, $\boldsymbol{v}_c = D_{\text{body}}(\boldsymbol{x}_c, \boldsymbol{d}_c)$, where $(\boldsymbol{x}_c, \boldsymbol{d}_c)$ is the centroid of cluster c and D_{body} is from the pre-trained discriminator. For the data sample $(\boldsymbol{x}, \boldsymbol{d})$ in the target city, we also get its current embedding $\boldsymbol{v}$ using D'_{body}, which is the discriminator being fine-tuned. We minimize the distance between the two embeddings, and thus realize the flexible fine-tuning.

The final objective during the fine-tuning process is as follows:

$$V(G, D) = \mathcal{L}_{cGAN}(G, D) + \alpha \mathcal{L}_{L1}(G) - \beta \mathcal{L}_{\text{c}}(D), \tag{3.11}$$

and the optimization process is $\min_G \max_D V(G, D)$.

The detailed fine-tuning algorithm is as Alg 10. The parameters of STrans-GAN are fine-tuned using Eq 3.12 and Eq 3.13. After fine-tuning, the generator G can be used for traffic estimation in any region of the target city.

$$\begin{aligned} V(\boldsymbol{\theta}_D) &= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D(\boldsymbol{x}, \boldsymbol{d})] \\ &+ \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z}, \boldsymbol{d}), \boldsymbol{d})] - \beta \mathcal{L}_{\text{c}}(D), \\ \boldsymbol{\theta}_D &= \boldsymbol{\theta}_D + \eta \nabla_{\boldsymbol{\theta}_D} V(\boldsymbol{\theta}_D). \end{aligned} \tag{3.12}$$

$$\begin{aligned} V(\boldsymbol{\theta}_G) &= \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z}, \boldsymbol{d}), \boldsymbol{d}))] + \alpha \mathcal{L}_{L1}(G), \\ \boldsymbol{\theta}_G &= \boldsymbol{\theta}_G - \eta \nabla_{\boldsymbol{\theta}_G} V(\boldsymbol{\theta}_G). \end{aligned} \tag{3.13}$$

Table 3.1: Dataset descriptions.

City	Data	Timespan	City size	Data size
Shenzhen	Speed	07/01/16-12/31/16	40×50	(155520,5,5)
	Inflow			
	Demand			
Harbin	Speed	07/01/15-12/31/15	40×50	(151440,5,5)
	Inflow			
	Demand			
Chengdu	Speed	10/01/16-10/31/16	20×20	(4784,5,5)
	Inflow			
	Demand			
Xi'An	Speed	10/01/16-10/31/16	20×20	(4368,5,5)
	Inflow			
	Demand			

Table 3.2: Performance on experiment 1: traffic speed and taxi inflow estimation in Xi'An.

Methods		Smoothing	cGAN	Curb-GAN	TrafficGAN	Single-TL	Multi-TL	RegionTrans	MetaST	**STrans-GAN**
Speed	RMSE	14.062±0.471	19.255±0.726	15.748±0.562	14.783±0.578	13.783±0.242	11.702±0.685	14.697±0.281	14.771±0.572	**5.881±0.133**
	MAPE	0.524±0.019	0.818±0.028	0.590±0.023	0.596±0.017	0.381±0.012	0.348±0.017	0.557±0.021	0.566±0.015	**0.241±0.007**
Inflow	RMSE	144.335±4.258	102.938±4.213	188.368±6.244	101.168±3.321	112.379±4.321	105.986±5.299	137.457±5.928	118.219±4.576	**79.362±2.784**
	MAPE	2.796±0.021	0.912±0.068	1.589±0.158	1.837±0.110	0.787±0.062	0.878±0.044	3.258±0.352	0.860±0.025	**0.445±0.016**

3.1.4 Evaluation

3.1.4.1 Dataset and Experiment Descriptions

Dataset Descriptions. In our experiments, we collect traffic data including travel demand, taxi inflow and traffic speed from multiple cities (*i.e.*, Chengdu, Xi'An, Shenzhen and Harbin) to evaluate our STrans-GAN. The detailed information of datasets is shown in Table 3.1.

In each city, with map gridding method, the whole city is partitioned into equal-sized grid cells, and we collect three different traffic datasets, *i.e.*, traffic speed, taxi inflow and travel demand. During each time slot (*i.e.*, one hour), traffic speed is the average speed extracted from taxis GPS records; taxi inflow indicates the total number of taxis that get into a grid cell; and travel demand calculates the total number of taxi pickup events within a grid cell. In this study, we use the demand for taxis to represent travel demand, and many studies have shown the effectiveness of using taxi demand to represent travel

Table 3.3: Performance on experiment 2: traffic speed and taxi inflow estimation in Chengdu.

Methods		Smoothing	cGAN	Curb-GAN	TrafficGAN	Single-TL	Multi-TL	RegionTrans	MetaST	**STrans-GAN**
Speed	RMSE	15.879±0.262	32.650±0.821	23.098±0.679	20.738±0.592	22.229±0.671	23.322±0.542	18.991±0.312	25.393±0.524	**11.763±0.231**
	MAPE	0.758±0.037	1.655±0.048	0.890±0.028	0.911±0.016	0.395±0.022	0.274±0.032	0.679±0.032	0.947±0.034	**0.176±0.011**
Inflow	RMSE	241.975±6.634	449.551±8.946	223.745±7.267	378.909±8.918	503.079±12.982	278.867±6.213	588.33±10.192	240.394±7.214	**95.597±4.423**
	MAPE	8.198±0.625	12.833±0.827	5.653±0.261	12.740±0.672	5.181±0.337	4.681±0.224	12.754±0.482	5.729±0.121	**1.181±0.033**

demand [1, 2, 14, 15]. More details about the dataset in each city are as follows:

- **Shenzhen.** The traffic data collected in Shenzhen, China is from Jul 1st to Dec 31st, 2016. The whole city is partitioned into 40×50 grid cells with a side-length $l_1 = 0.0084°$ in latitude and $l_2 = 0.0126°$ in longitude, and each region is formed by 5×5 grid cells. There are 155,520 region-wise traffic distributions in total, and each traffic distribution is a 5×5 matrix.

- **Harbin.** The traffic data collected in Harbin, China is from Jul 1st to Dec 31st, 2015. Harbin City is partitioned into 40×50 grid cells with a side-length $l_1 = 0.0084°$ in latitude and $l_2 = 0.0126°$ in longitude, and each region is formed by 5×5 grid cells. There are 151,440 region-wise traffic distributions in total.

- **Chengdu** The traffic data is collected from only one district of Chengdu, China, which is partitioned into 20×20 grid cells with multiple 5×5 regions, each grid cell has a side-length $l_1 = 0.0038°$ in latitude and $l_2 = 0.0045°$ in longitude. The data time span is from Oct 1st to Oct 31st, 2016. The total number of traffic distributions is 4,784.

- **Xi'An.** Similar to Chengdu, the traffic data in Xi'An is also collected from one district, which is also partitioned into 20×20 grid cells with a side-length $l_1 = 0.0041°$ in latitude and $l_2 = 0.0048°$ in longitude. The data time span is from Oct 1st to Oct 31st, 2016. The total number of traffic distributions is 4,784.

Experiment Descriptions. In our evaluation section, we conduct 2 different experiments to prove the effectiveness of our STrans-GAN:

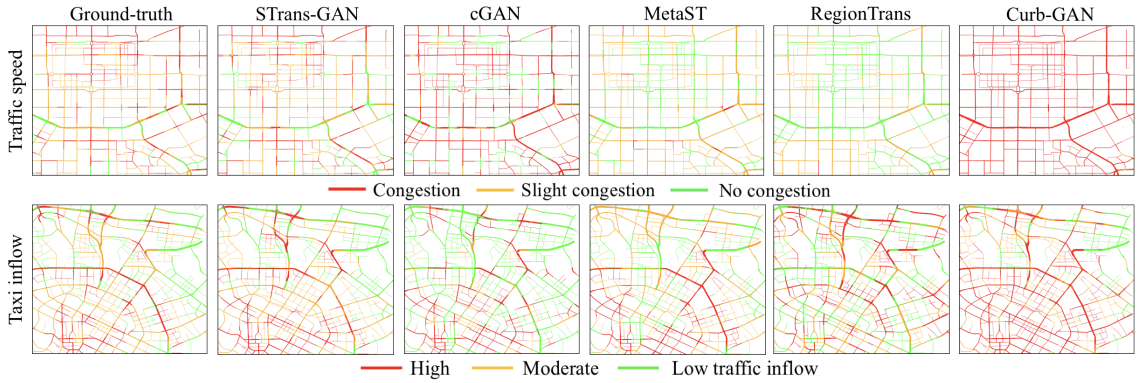


Figure 3.4: Visualizations of (1st row) traffic speed estimation in Xi'An & (2nd row) taxi inflow estimation in Chengdu.

- **Experiment 1: traffic speed and taxi inflow estimation in Xi'An.** In this experiment, we treat Shenzhen, Harbin and Chengdu as source cities, and view Xi'An as the target city. Given new travel demands of different regions in Xi'An, we aim to estimate the corresponding traffic speed and taxi inflow. Besides, 70% of data from Xi'An is used for fine-tuning, the remaining 30% is used for testing.

- **Experiment 2: traffic speed and taxi inflow estimation in Chengdu.** In this experiment, we treat Shenzhen, Harbin and Xi'An as source cities, and view Chengdu as the target city. Given new travel demands of different regions in Chengdu, we aim to estimate both traffic speed and taxi inflow. Similarly, we also use 70% of data from Chengdu for fine-tuning, and the remaining for testing.

3.1.4.2 Baselines

To verify the existing traffic estimation methods cannot produce good estimations when the target city has data scarcity problem, we compare our model with the state-of-the-art traffic estimation models:

- **Spatial Smoothing [34].** Given one target region and its travel demand, this method selects the traffic distributions from nearby regions which have similar travel demands

and then computes an average traffic distribution as the final estimation.

- **cGAN [33].** The conditional GAN applies convolutional layers inside both generator and discriminator. We use travel demands as conditions to estimate the corresponding traffic distributions.

- **Curb-GAN [2, 70].** Curb-GAN applies self-attention and convolutional layers to deal with sequential traffic data generation problem. The generator is trained with the adversarial loss and L1 loss together.

- **TrafficGAN [1, 66]** TrafficGAN applies dynamic convolutional layers inside both generator and discriminator, and the generator is trained using both adversarial loss and L2 loss.

Besides, we compare our STrans-GAN with state-of-the-art transfer leaning methods including both single-source and multi-source transfer learning methods:

- **Multi-TL** Multi-source transfer learning uses the same architecture as STrans-GAN, but it is trained with simply mixed data from all source cities without clustering.

- **Single-TL** Single-source transfer learning uses the same architecture as STrans-GAN, but only has one source city for training.

- **RegionTrans [74]** This model targets time-series traffic prediction problem and only allows knowledge transfer from one source city to the target.

- **MetaST [79]** This model supports knowledge transfer from multiple cities, which treats each city as a task without considering different traffic patterns and directly applies MAML [85].

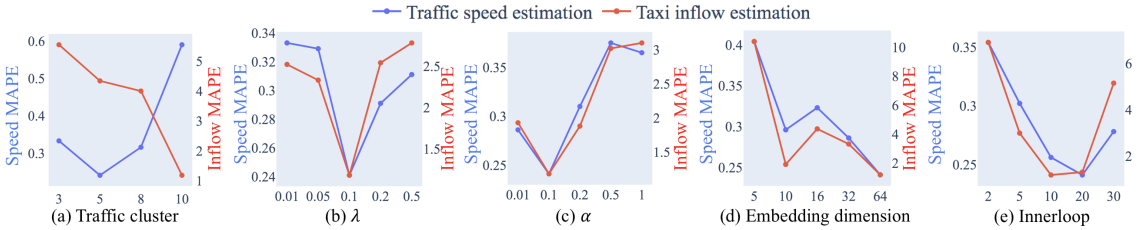


Figure 3.5: Hyper-parameters studies on traffic speed estimation (Xi'An) and taxi inflow estimation (Chengdu).

3.1.4.3 Evaluation Metrics

In our experiments, mean absolute percentage error (MAPE) and rooted mean square error (RMSE) are used to evaluate our model:

$$\text{MAPE} = \frac{1}{n_s}\sum_{i=1}^{n_s} |y_i - \hat{y}_i| / y_i, \text{RMSE} = \sqrt{\frac{1}{n_s}\sum_{i=1}^{n_s} (y_i - \hat{y}_i)^2}, \tag{3.14}$$

where n_s is the total number of grid cells in the target region, y_i is the ground-truth traffic status observed in one grid cell s_i, and $\hat{y}_i$ is the corresponding estimated value.

3.1.4.4 Experimental Settings

For both experiments, we use data from three different source cities for pre-training, and select 70% of the data from the target city for fine-tuning and use the remaining 30% for testing. All models are updated using Adam optimizer [56]. The learning rate is set to 2×10^{-5}. The batch size is 32. Besides, we set $\alpha = 0.1$, $\beta = 0.1$, $\lambda = 0.1$. The detailed structure of STrans-GAN in our experiments is as follows: the generator G contains 4 transposed convolutional layers with kernel sizes $\{5, 5, 5, 5\}$ and output feature dimensions $\{1024, 128, 64, 1\}$; the discriminator D_{body} includes four convolutional layers with kernel sizes $\{5, 5, 5, 5\}$ and output channels $\{64, 128, 1024, 1024\}$. And the D_{head} only has one single fully-connect layer.

Table 3.4: Ablation Study: traffic speed and taxi inflow estimation in Xi'An.

Methods		STrans-GAN$_c$	STrans-GAN$_p$	STrans-GAN$_f$	STrans-GAN$_r$	**STrans-GAN**
Speed	RMSE	11.702± 0.685	19.88±1.295	18.624±0.899	15.354±0.825	**5.881±0.133**
	MAPE	0.348±0.017	0.899±0.055	0.754±0.067	0.684±0.043	**0.241±0.007**
Inflow	RMSE	105.986±5.299	107.266±5.837	113.265±7.399	94.682±4.829	**79.362±2.784**
	MAPE	0.878±0.044	0.897±0.079	0.925±0.081	0.681±0.032	**0.445±0.016**

3.1.4.5 Experimental Results

Estimation Performance. In this part, we provide the evaluation statistics for our STrans-GAN and all baselines in both experiment. All deep models are trained and fine-tuned three times, and during testing time, we estimate the traffic for 16 different regions in the target city, and finally calculate the statistics using Eq 3.14. Based on the performance shown in Table 3.2 and Table 3.3, we have the following observations.

In both experiments, the traditional smoothing method has very high MAPE, which means the estimation at each grid cell is not accurate due to not considering local traffic patterns; other state-of-the-art traffic estimation models (including cGAN, TrafficGAN and Curb-GAN) do not provide good estimation results due to the lack of training samples. Besides, the single-source transfer learning methods (*i.e.*, RegionTrans and Single-TL) have low estimation quality, since these methods only take one city as source city, once the source city doesn't present similar traffic patterns to the target city, these methods would fail. For the multi-source transfer learning methods (*i.e.*, MetaST and Multi-TL), they cannot detect different traffic patterns in each source city, and thus they are unable to guarantee the model performance. Our STrans-GAN outperforms all baseline models in both experiments since it takes diverse traffic patterns into consideration and learns a good initialization from multiple source cities, and it is also successfully adapted to different scenarios in the target city.

Estimation Visualization. To further validate the effectiveness of our STrans-GAN, for both target cities including Xi'An and Chengdu, we present the visualizations of the ground-truth and the estimated traffic distributions over the road networks.

As shown in Figure 3.4, we visualize the traffic distributions including the ground-

truth and the estimation results generated by our STrans-GAN and some competitive baselines (*i.e.*, cGAN, Curb-GAN, metaST and RegionTrans) in Xi'An City (the 1st row) and Chengdu City (the 2nd row). Obviously, in both experiments, cGAN and Curb-GAN cannot capture the traffic patterns very well due to the lack of data. MetaST and RegionTrans also present low quality estimations, they cannot identify different traffic patterns from source cities and fail to transfer useful knowledge to different target regions in the target city. In contrast, our STrans-GAN generates reasonable traffic distributions which are close to the ground-truth, and our STrans-GAN is able to successfully extract traffic knowledge from multiple source cities and adapt to the target city in a flexible way.

3.1.4.6 Hyper-parameter Studies

In our STrans-GAN, there are many hyper-parameters which may influence the model performance. In this section, we study how hyper-parameters influence the estimation performance of our STrans-GAN. The evaluated hyper-parameters includes the number of traffic clusters, embedding dimension, innerloop, λ and α.

From the results shown in Fig 4.7(a), we find in different target cities, the ideal number of traffic clusters varies since different cities would present different traffic patterns. In Xi'An City, the ideal number of traffic clusters is 5, in Chengdu City, the best number of traffic clusters is 10, which indicates Chengdu City has more complex traffic patterns compared to the traffic in Xi'An City.

In Fig 4.7(b), when updating the global-initialized STrans-GAN parameters $\boldsymbol{\theta}_G$ and $\boldsymbol{\theta}_D$ during pre-training process, the step size λ matters. Too small or large λ hinders convergence. For both experiments, the ideal step size λ should be 0.1.

In Fig 4.7(c), we can find the estimation performance is sensitive to the value of α, α should be chosen based on the adversarial loss scale to ensure the whole loss scale keeps the same, in our experiments, the best choice of α is 0.1.

In Fig 4.7(d), we investigate how the embedding dimension influences the estimation performance. We find in both experiments, larger embedding dimension provides better estimation results, which indicates that larger embedding dimension can better preserve traffic patterns and potentially improve the estimation performance.

In Fig 4.7(e), the value of innerloop in Alg 9 is the number of steps when we adapt to each traffic cluster, and we study how the number of innerloop affects performance. It is obvious that too small or too large innerloop value usually leads to longer time to converge. In our experiments, the ideal number of innerloop should be 10 or 20.

Ablation Studies. Our STrans-GAN is composed of multiple components, including traffic clustering, pre-training, fine-tuning, and cluster matching regularizer, in this section, to verify the contribution of each component in our model, we present ablation studies.

As shown in Table 3.4, we estimate traffic speed and taxi inflow with Xi'An City as the target city, and compare our STrans-GAN with STrans-GAN$_p$, STrans-GAN$_c$, STrans-GAN$_r$, STrans-GAN$_f$. STrans-GAN$_c$ removes the traffic clustering module, STrans-GAN$_p$ removes the pre-training part. In STrans-GAN$_f$, fine-tuning process is removed. And in STrans-GAN$_r$, the clustering matching regularizer is removed. Apparently, if pre-training module is removed in STrans-GAN$_p$, the cluster matching regularizer cannot work without a pre-trained model, and the limited data in the target city cannot support GAN training; if fine-tuning module is removed, the model cannot directly estimate the traffic in the target city without access to any data from the target; if the cluster matching regularizer is removed, the traffic clustering does not contribute to the fine-tuning process at all. Thus, each component in our Strans-GAN is important and contributes to the final estimation performance.

4

Learning Human Driving Strategies

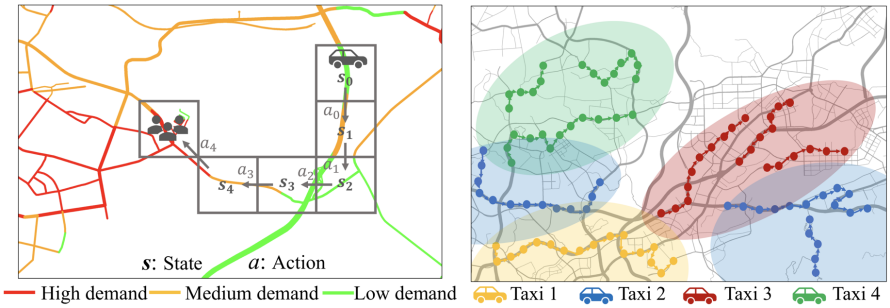
4.1 Spatial-Temporal Meta-GAIL

4.1.1 Overview

4.1.1.1 Introduction

In human urban decision-making processes (*e.g.*, taxi drivers' passenger-seeking processes as shown in Figure 4.1(a), travelers' transit mode choices, *etc.*), human agents devise their own "strategies" to optimize their objectives (*e.g.*, maximizing revenue, minimizing travel time, *etc.*). These strategies are usually implicit to observers and even the agents themselves, which govern the daily mobility patterns of the human agents. *Human urban strategy analysis* aims at extracting and understanding how the human decisions are made using the observed human-generated spatial-temporal urban data (*e.g.*, GPS trajectories of taxis and personal vehicles, passengers' trip data on buses and trains, *etc.*). This is an important problem in many urban intelligence scenarios such as ride-sharing vehicle dispatching, public transportation management, and autonomous driving, *etc.*

Challenges. The human urban strategy analysis problem is challenging due to the fol-



(a) Taxi driver's passenger-seeking trajectory (b) Data scarcity and heterogeneity

Figure 4.1: Examples of taxi driver's decision-making process (left), data scarcity and heterogeneity challenges (right).

lowing two reasons. Firstly, learning from observations usually requires large amounts of training data from the agent being studied. However, in many urban cases, it is hard to collect abundant mobility data from a single human agent (*i.e.*, data scarcity challenge). Second, data collected from urban scenarios is often heterogeneous, meaning that it records behaviors of many different expert agents following various urban strategies (*i.e.*, data heterogeneity challenge, See Figure 4.1(b) for example). This makes it even harder to extract explicit and reliable strategies of a target agent.

Prior works. Over the last few years, many imitation learning algorithms have been proposed to conduct the human urban strategy analysis. For example, Ho et al. proposed generative adversarial imitation learning (GAIL) [86] which can successfully learn human decision-making strategies and accurately mimic human behaviors in various scenarios using deep neural networks (DNNs). Pan et al. proposed an Explainable xGAIL [87] to demonstrate the learning processes of GAIL in many real world cases. Zhang et al. extended the standard GAIL to conditional GAIL (cGAIL) [88] to unveil taxi drivers' driving policies by transferring knowledge across taxi drivers. Moreover, TrajGAIL [89] proposed by Zhang et al. incorporates the self-attention mechanism into GAIL to capture the long-term decision dependencies and learn the human decision strategies.

However, all these prior methods fail to properly address the above aforementioned

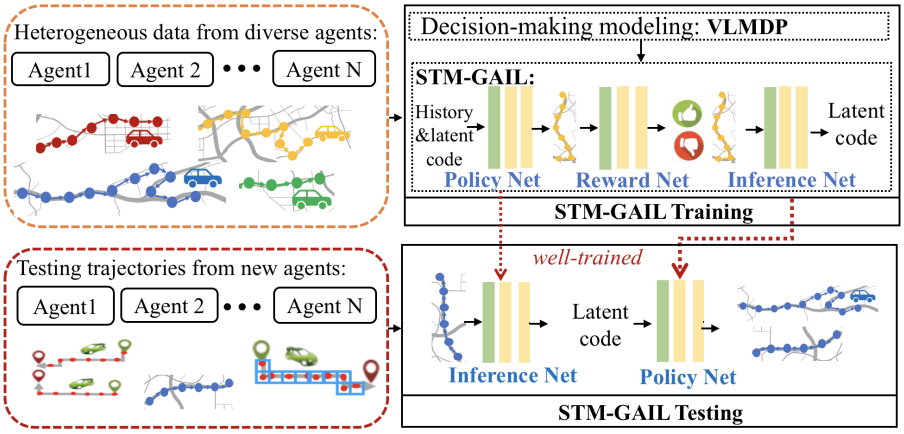


Figure 4.2: Solution framework.

challenges at the same time. These methods learn human urban strategies from scratch and require a large amount of historical behaviors of a single human agent, aiming to correctly infer her urban strategies. In case of data scarcity (*i.e.*, each agent has limited observations) and heterogeneity (*i.e.*, having mixed observations from many agents), such as inferring the driving strategies of a new taxi driver based on all drivers' trajectories, all of these methods may fail.

Other works including InfoGAIL [90] proposed by Li et at. and MetaIL [91] proposed by Yu et at. aiming to learn a well-generalized policy which can be adapted to new related tasks. Unfortunately, they only focus on learning different skills from visual demonstrations and do not consider diverse urban spatial-temporal conditions, and thus cannot successfully learn various human urban strategies in complex urban scenarios.

Contributions. To solve the human urban strategy analysis problem in case of data scarcity and data heterogeneity, in this work, a novel learning paradigm —Spatial-Temporal Meta-GAIL (STM-GAIL) is proposed, which can successfully learn diverse human urban strategies from heterogeneous human-generated spatial-temporal urban data. Our solution framework is shown in Figure 2. STM-GAIL contains three model components including a policy network, a reward network and an inference network. It models the

human urban decision-making processes as variable length Markov decision processes (VLMDPs). STM-GAIL learns diverse human urban strategies from a meta-learning perspective, which exploits the structural similarity among a distribution of human experts' urban strategies and optimizes for rapid adaptation to a new strategy with a single trajectory.

Our main contributions can be summarized as follows:

- We make the first attempt to learn diverse human urban decision-making strategies in case of data scarcity and data heterogeneity, and propose the Spatial-Temporal Meta-GAIL (STM-GAIL), which incorporates the spatial-temporal dependencies of human decisions into GAIL framework by modeling the human urban decision-making processes as variable length Markov decision processes (VLMDPs) and taking the surrounding spatial feature patterns (*e.g.*, traffic volume patterns, travel demand patterns, *etc.*) as part of the states.

- STM-GAIL learns diverse human strategies from the meta-learning perspective, novel objective, architecture and algorithms are designed. In STM-GAIL, an inference network is designed on top of the standard GAIL, which infers the latent variables of diverse human strategies in an unsupervised way by maximizing the mutual information between the latent space and trajectories. STM-GAIL can be generalized to a new human expert's urban strategy with a single trajectory.

- Extensive experiments on real-world human-generated spatial-temporal dataset (*i.e.*, taxi trajectory dataset representing the taxi drivers' passenger-seeking processes) are performed to validate the effectiveness of our STM-GAIL. The experimental results show that our STM-GAIL has significant improvement compared to state-of-the-art baselines when learning human urban strategies.

Table 4.1: Notations

Notations	Descriptions
$\mathcal{C} = \{c_{ij}\}$	Grid cells within a city
$\mathcal{S} = \{\boldsymbol{s}_t\}$	State set
$\mathcal{A} = \{a_t\}$	Action set
$\mathcal{T} = \{\tau\}$	Trajectory set
$\mathcal{H} = \{h\}$	History set
$c \sim p(c)$	latent vector representing a specific strategy
$\pi(a \mid \boldsymbol{s}, h, c)$	Policy network
$\pi_E(a \mid \boldsymbol{s}, h, c)$	Empirical policy from the data
$r(\boldsymbol{s}, a \mid h, c)$	Reward network
$Q(c \mid \tau)$	Inference network
$\boldsymbol{\theta}$, $\boldsymbol{\omega}$, $\boldsymbol{\psi}$	Parameters of π, r, Q

4.1.1.2 Preliminaries

Human-generated spatial-temporal urban data is collected from expert human agents to learn human urban decision-making strategies. In general, human-generated spatial-temporal urban data is a set of human mobility trajectories (*e.g.*, taxi GPS trajectories) which contains sequences of states and actions. In this section, we list the notations in Table 5.1, introduce some definitions, and formally define our problem.

Definition 1 (Grid cells). A city is partitioned into $m_1 \times m_2$ grid cells, each grid cell has equal side-length in latitude and longitude. The set of grid cells of a city is defined as $\mathcal{C} = \{c_{ij}\}$, where $1 \leq i \leq m_1$ and $1 \leq j \leq m_2$. Each grid cell is associated with a set of features (*e.g.*, traffic speed, traffic volume, *etc.*) indicating the current status of the grid cell [1, 2, 3].

Definition 2 (States). A state at time t is defined as a multi-dimensional tensor $\boldsymbol{s}_t \in \mathbb{R}^{m \times r \times r}$, which is composed of m different feature maps $\boldsymbol{d} \in \mathbb{R}^{r \times r}$, each element $d_c \in \mathbb{R}$ inside a feature map $\boldsymbol{d}$ indicates a feature (*e.g.*, latitude, longitude, time of the day, traffic speed, travel demand, *etc.*) of a specific grid cell c at time t. The set of states is defined as $\mathcal{S} = \{\boldsymbol{s}_t\}$.

In most of the existing works, a state only characterizes the current status (*e.g.*, lati-

tude, longitude, time of the day, traffic speed, traffic volume, *etc.*) of a specific grid cell. However, the human agents would also consider the status of the surrounding area when making urban decisions. Thus, in most of the urban scenarios, a state should characterize the status of both the current grid cell and its neighboring $r \times r$ grid cells as illustrated in Figure 4.3(a).

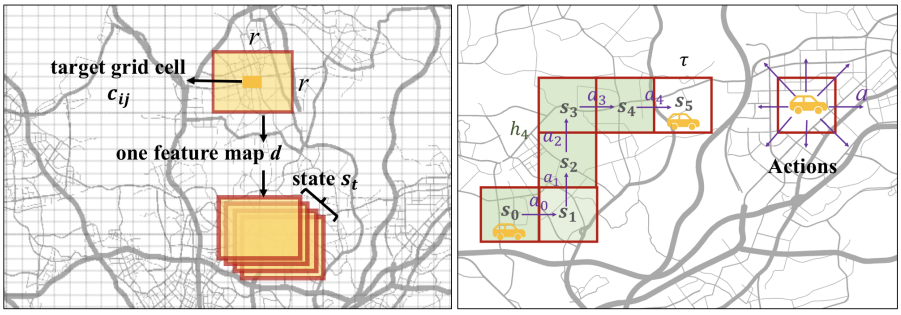
Definition 3 (Actions). An action a_t is a decision made by a human agent at state $\boldsymbol{s}_t$, which is governed by a specific urban strategy. By following an action a_t, the human agent transits from the current state $\boldsymbol{s}_t$ to the next state $\boldsymbol{s}_{t+1}$ as shown in Figure 4.3(b). The set of actions is defined as $\mathcal{A} = \{a_t\}$.

Definition 4 (Trajectories and History). A trajectory τ is a sequence of states and actions that a human agent traverses and takes when completing a task, *i.e.*, $\tau = (\boldsymbol{s}_0, a_0, \cdots, \boldsymbol{s}_T, a_T)$. The history of a trajectory τ at time step t includes all states and actions prior to t, *i.e.*, $h_{t-1} = (\boldsymbol{s}_0, a_0, \cdots, \boldsymbol{s}_{t-1}, a_{t-1})$. The set of trajectories is defined as $\mathcal{T} = \{\tau\}$. The set of histories is defined as $\mathcal{H} = \{h\}$.

Definition 5 (Policy). The policy function $\pi : \mathcal{S} \times \mathcal{H} \mapsto [0, 1]$ controls what action to perform in each state, which is a probability distribution defined as $\pi(a_t \mid \boldsymbol{s}_t, h_{t-1})$ indicating the probabilities of choosing different actions given the current state $\boldsymbol{s}_t$ and the history h_{t-1}.

Definition 6 (Reward). The reward function is defined as $r : \mathcal{S} \times \mathcal{A} \times \mathcal{H} \mapsto \mathbb{R}$, *i.e.*, $r(\boldsymbol{s}_t, a_t \mid h_{t-1})$, which provides a numerical score based on a state $\boldsymbol{s}_t$ and an action a_t given the history h_{t-1}, and incentivizes a human agent to achieve a goal in a task.

Problem Statement. Given a set of heterogeneous trajectories $\mathcal{T}$ generated by a wide range of expert human agents, and a few trajectories $\tilde{\mathcal{T}}$ collected from new experts, we aim to learn the urban decision-making strategies of new experts, *i.e.*, the policy function $\pi(a \mid \boldsymbol{s}, h)$.



(a) Map gridding and state (b) Action, trajectory and history

Figure 4.3: Definition illustrations.

4.1.2 Related Work

In this section, we summarize the literature works from two related areas: 1) imitation learning, and 2) meta learning.

Imitation Learning. Imitation learning also known as learning from demonstrations aims to learn the policies from expert demonstrations. Most of the imitation learning methods model the decision making processes as Markov Decision Processes (MDPs) [92, 93, 94, 95]. For example, GAIL [86] borrows the generative adversarial networks (GANs) framework and applies two neural networks as policy net and reward net to learn the policies of experts from demonstrations. Many works extend the GAIL framework to diverse urban applications, for example, Kuefler et al. try to use GAIL based model to imitate the expert drivers' behaviors in autonomous driving [96]. Zhang et al. extend the standard GAIL to conditional GAIL (cGAIL) [88] to unveil taxi drivers' policies by transferring knowledge across taxi drivers and locations. To deal with the spatial-temporal urban strategy learning problem, only a few works model the decision making processes as variable length Markov decision processes (VLMDPs) to capture the long-term decision dependencies. For example, TrajGAIL [89] incorporates the self-attention mechanism into GAIL framework and learns the human decision-making strategies in the VLMDP setup. However, all these methods learn strategies from scratch and require a

large amount of demonstrations, and cannot learn the diverse urban strategies directly.

Meta Learning. Meta learning [82, 85, 97] tries to learn a generalized model from training tasks which can be fast adapted into new related tasks with a few samples. Meta learning has been applied to many areas including supervised/unsupervised learning and imitation learning. For example, in meta imitation learning, many prior works focus on learning diverse tasks from mixed experts' demonstrations [98, 99, 100]. Moreover, one-shot imitation learning [91, 101, 102, 103] demonstrates impressive results on learning new tasks using a single demonstration, however, it requires a large amount of training tasks and needs prior knowledge on the task distribution. All these works did not consider the uniqueness of learning urban strategies, and cannot successfully capture the spatial-temporal dependencies of human decisions. In this work, we propose Spatial-Temporal Meta GAIL (STM-GAIL) which incorporates the spatial-temporal decision dependencies into the GAIL framework and learns diverse urban decision-making strategies from heterogeneous human-generated spatial-temporal urban data.

4.1.3 Methodology

Built upon the standard generative adversarial imitation learning (GAIL [86]), we propose a novel STM-GAIL to learn diverse human urban strategies. STM-GAIL takes the spatial and temporal dependencies into consideration by incorporating the feature maps of surrounding areas into states and modeling the human decision-making processes as VLMDPs. Moreover, ConvLSTM [51] is applied to each model component (including the policy network, the reward network and the inference network) to better capture the spatial-temporal dependencies of a trajectory (See Section 4.1.3.1 and 5.1.3.3).

In addition, to tackle data scarcity and heterogeneity challenges, STM-GAIL learns diverse human urban strategies from the meta-learning perspective, an inference network is designed on top of the standard GAIL, which infers the latent variables of diverse

human strategies in an unsupervised way. STM-GAIL can be generalized to a new human urban strategy with a single trajectory (See Section 4.1.3.2, 5.1.3.3 and 5.1.3.4).

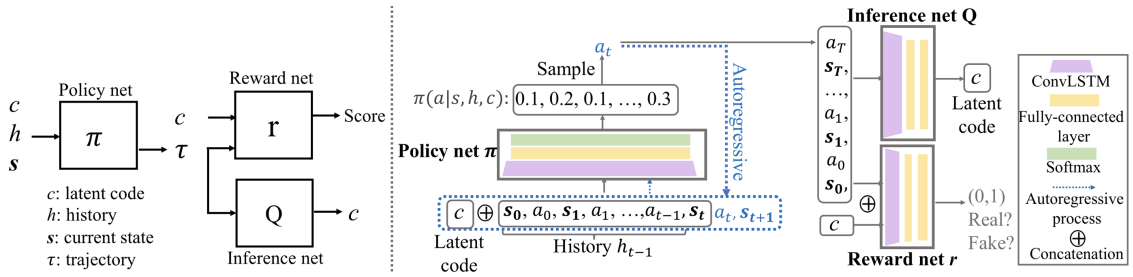
4.1.3.1 Modeling Human Sequential Decision-Making Process as VLMDP

Limitations of MDP. Over recent years, a large amount of works have focused on learning human decision-making strategies by modeling decision-making processes as Markov decision processes (MDPs) [104]. In general, MDP models have a strong Markov property assumption [105], namely, an agent makes an action a_t only based on the current state $\boldsymbol{s}_t$ instead of any prior states and actions (*i.e.*, history h_{t-1}). Thus, in MDPs, the policy and reward functions of a human agent should be $\pi(a_t \mid \boldsymbol{s}_t)$ and $r(\boldsymbol{s}_t, a_t)$, respectively, rather than $\pi(a_t \mid \boldsymbol{s}_t, h_{t-1})$ and $r(\boldsymbol{s}_t, a_t \mid h_{t-1})$.

However, in many urban scenarios, the Markov property does not hold, human agents would consider their previous states and decisions, as well as the surrounding environments when making future decisions. For example, as illustrated in Figure 4.1(a), when looking for a new passenger, a taxi driver's decisions of which direction to go not only depend on his current and previous locations, but also depend on the surrounding travel demand. Such spatial-temporal dependencies of human mobility are complicated and hard to capture when learning human urban strategies.

Human sequential decision-making processes as VLMDP.

To capture the long-term dependency of human decisions, we model the decision-making process as a variable length Markov decision process (VLMDP) [19], which includes an agent as the decision maker and an environment that interacts with the agent. A VLMDP is defined as a 5-tuple $\langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space; P denotes the transition function, *e.g.*, $P(\boldsymbol{s}_t|h_{t-1})$ is the transition probability of transiting to state $\boldsymbol{s}_t$ by following the history h_{t-1}; $r : \mathcal{S} \times \mathcal{A} \times \mathcal{H} \mapsto \mathbb{R}$ is the bounded reward function that outputs a reward value for a given state-action-history triple; $\gamma \in (0, 1]$



(a) STM-GAIL Overview (b) STM-GAIL Architecture

Figure 4.4: STM-GAIL structure.

is a discount factor. The initial states are determined by the distribution $p(\boldsymbol{s}_0) : \mathcal{S} \mapsto [0, 1]$. The actions are chosen through a stationary and stochastic policy $\pi : \mathcal{S} \times \mathcal{H} \mapsto [0, 1]$. A decision making process forms a trajectory $\tau = (\boldsymbol{s}_0, a_0, \cdots, \boldsymbol{s}_T, a_T)$, where T is the terminal time step.

In this work, we use expectation with respect to a policy π to denote the expectation with respect to the trajectories it generates. For instance, $\mathbb{E}_\pi[r(\boldsymbol{s}, a \mid h)] = \mathbb{E}_{\boldsymbol{s}_t, h_{t-1}, a_t \sim \pi}\left[\sum_{t=0}^{T} \gamma^t r\left(\boldsymbol{s}_t, a_t \mid h_{t-1}\right)\right]$ denotes the following sampling processes including $\boldsymbol{s}_0 \sim p(\boldsymbol{s}_0), a_t \sim \pi\left(\cdot \mid \boldsymbol{s}_t, h_{t-1}\right)$, and $\boldsymbol{s}_t \sim P\left(\boldsymbol{s}_t \mid h_{t-1}\right)$. Each agent aims to maximize its expected cumulative reward $\mathbb{E}_\pi[r(\boldsymbol{s}, a \mid h)]$ by optimizing the policy π.

Based on the variable length Markov decision process (VLMDP), we design the Spatial-Temporal Meta-GAIL (STM-GAIL) to learn diverse human urban strategies.

4.1.3.2 Objective

In many previous MDP works [86, 87, 88, 106], the human strategy learning problem can be modeled as a constrained optimization problem as below:

$$\begin{aligned} &\max_r \min_\pi : -H(\pi), \\ &\text{s.t.} : \mathbb{E}_\pi[r(\boldsymbol{s}, a)] = \mathbb{E}_{\pi_E}[r(\boldsymbol{s}, a)], \\ &\sum_{a \in \mathcal{A}} \pi(a \mid \boldsymbol{s}) = 1, \quad \forall \boldsymbol{s} \in \mathcal{S}. \end{aligned} \tag{4.1}$$

However, Eq 4.1 does not consider any temporal dependencies of decisions. In this work, to incorporate the long-term temporal dependencies of human decisions and adapt the human strategy learning problem to VLMDP, the problem is re-formulated as Eq 4.2 [89]:

$$\begin{aligned} &\max_{r} \min_{\pi} : -H(\pi), \\ &\quad \text{s.t.} : \mathbb{E}_{\pi}[r(\boldsymbol{s}, a \mid h)] = \mathbb{E}_{\pi_E}[r(\boldsymbol{s}, a \mid h)], \\ &\quad \sum_{a \in \mathcal{A}} \pi(a \mid \boldsymbol{s}, h) = 1, \quad \forall \boldsymbol{s} \in \mathcal{S}. \end{aligned} \tag{4.2}$$

In Eq 4.2, $H(\pi)$ is a γ-discounted causal entropy, which measures the uncertainty of a policy distribution $\pi(a \mid \boldsymbol{s}, h)$, *i.e.*, $H(\pi) = \sum_{t=0}^{T} \sum_{h_t} \gamma^t \pi\left(a_t \mid \boldsymbol{s}_t, h_{t-1}\right) \log \pi\left(a_t \mid \boldsymbol{s}_t, h_{t-1}\right)$. π_E is the empirical policy observed from the collected human expert's mobility data. Eq 4.2 aims to find the policy $\pi(a \mid \boldsymbol{s}, h)$ with maximum causal entropy $H(\pi)$, and find the reward function $r(\boldsymbol{s}, a \mid h)$ such that the expected reward of a trajectory under π matches that under the empirical policy π_E.

To solve the human strategy learning problem defined in Eq 4.2, Zhang et.al [89] prove it is equivalent to solving a min-max problem as Eq 4.3:

$$\min_{\pi \in \Pi} \max_{r} -\lambda_1 H(\pi) + \mathbb{E}_{\pi_E}[\log(r(\boldsymbol{s}, a \mid h))] + \mathbb{E}_{\pi}[\log(1 - r(\boldsymbol{s}, a \mid h))]. \tag{4.3}$$

Apparently, Eq 4.3 is similar to the objective of generative adversarial networks (GANs) [29, 33, 52], and it is natural to employ the GAN framework, where the policy function π and the reward function r can be viewed as a generator and a discriminator, respectively.

Even though Eq 4.3 can capture the spatial-temporal dependencies of decisions very well, it cannot deal with the data scarcity and heterogeneity problem. Eq 4.3 can learn a single human expert's urban strategy with access to abundant trajectories, once we cannot collect enough historical trajectories from the human expert, this method would fail. Moreover, when facing the data heterogeneity problem, namely, the trajectories are col-

lected from different human experts, Eq 4.3 would simply assume all trajectories are produced by one human expert and fail to learn different urban strategies from the dataset.

Thus, to deal with the data scarcity and heterogeneity problems, we introduce a latent variable c to our policy function and reward function, *i.e.*, $\pi(a|\boldsymbol{s}, h, c)$ and $r(\boldsymbol{s}, a \mid h, c)$, respectively. In general, $c \sim p(c)$ would be a latent vector representing a specific strategy of a human expert in the latent space. To enable the latent variable c to identify different strategies, a constraint on c should be added to Eq 4.3. Inspired by InfoGAN [62], where the latent variable is incorporated to enable disentangled representations by discovering the salient semantic features of the data distribution, we propose to add a mutual information regularizer to Eq 4.3 to encourage strong connections between c and the generated human trajectories. The mutual information between the latent variable and trajectories is denoted as $I(c;\tau)$, the objective with the mutual information regularizer is as follows:

$$\begin{aligned}\min_{\pi\in\Pi}\max_{r}\mathbb{E}_{\pi_E}[\log(r(\boldsymbol{s}, a \mid h, c))] + \mathbb{E}_{\pi}[\log(1 - r(\boldsymbol{s}, a \mid h, c))] \\ - \lambda_1 H(\pi) - \lambda_2 I(c;\tau).\end{aligned} \tag{4.4}$$

In Eq 4.4, the latent variable c helps to identify different strategies in a heterogeneous dataset and also enable fast generalization to new strategies with few samples. However, in practice, it is hard to directly maximize the mutual information $I(c;\tau)$ without the access to the posterior distribution $P(c|\tau)$. Instead, we calculate the variational lower bound [62, 107] of $I(c;\tau)$ and use an auxiliary distribution $Q(c|\tau)$ to approximate the

true posterior $P(c|\tau)$:

$$
\begin{aligned}
I(c;\tau) &= H(c) - H(c \mid \tau) \\
&= \mathbb{E}_{\tau\sim\pi(\cdot|\boldsymbol{s},h,c),c'\sim P(c|\tau)}\left[\log P\left(c' \mid \tau\right)\right] + H(c) \\
&= \mathbb{E}_{\tau\sim\pi(\cdot|\boldsymbol{s},h,c)}[\underbrace{D_{\mathrm{KL}}(P(\cdot \mid \tau)\|Q(\cdot \mid \tau))}_{\geq 0} \\
&+ \mathbb{E}_{c'\sim P(c|\tau)}[\log Q(c' \mid \tau)]] + H(c) \\
&\geq \mathbb{E}_{c\sim p(c),\tau\sim\pi(\cdot|\boldsymbol{s},h,c)}[\log Q(c \mid \tau)] + H(c) \\
&= L_I(\pi, Q),
\end{aligned}
\tag{4.5}
$$

where $p(c)$ is a prior distribution, Q is the auxiliary distribution, and we can treat Q as an inference neural network, which uses τ to infer c. As a result, the final objective for our Spatial-Temporal Meta-GAIL (STM-GAIL) is as Eq 5.7:

$$
\begin{aligned}
\min_{\pi,Q}\max_{r}&\mathbb{E}_{\pi_E}[\log(r(\boldsymbol{s}, a \mid h, c))] + \mathbb{E}_{\pi}[\log(1 - r(\boldsymbol{s}, a \mid h, c))] \\
&- \lambda_1 H(\pi) - \lambda_2 L_I(\pi, Q).
\end{aligned}
\tag{4.6}
$$

4.1.3.3 Model Architecture

In our final objective Eq 5.7, a policy network π, a reward network r and an inference network Q are required. Figure 5.5 shows the detailed architecture of STM-GAIL, which applies ConvLSTM [51] inside each model component to better capture the spatial-temporal dependencies of human decisions in a trajectory.

The policy network π outputs an action distribution $\pi(a_t|\boldsymbol{s}_t, h_{t-1}, c)$ based on the current state $\boldsymbol{s}_t$, the history h_{t-1} and the latent vector c. A specific action a_t will be sampled from the distribution, *i.e.*, $a_t \sim \pi(a_t|\boldsymbol{s}_t, h_{t-1}, c)$. Given the sampled action a_t, the next state $\boldsymbol{s}_{t+1}$ is directly provided by the environment (through the transition function

$P(\boldsymbol{s}_{t+1}|h_t)$[1]), which is combined with the extended history h_t and latent vector c as the new input of the policy network, *i.e.*, $a_{t+1} \sim \pi(a_{t+1}|\boldsymbol{s}_{t+1}, h_t, c)$. Thus, the policy network works in an auto-regressive way. Inside the policy network π, the current state $\boldsymbol{s}_t$ and the latent vector c are concatenated together and pass a ConvLSTM, the history h_{t-1} is stored within the hidden states [51] of ConvLSTM. The output of the ConvLSTM passes a fully-connected layer and a softmax function [109] to get the probabilities of choosing different actions.

The reward network r can be viewed as a discriminator, which aims to distinguish the positive data from the negative data by giving high scores if the input τ is collected from expert human agents, and giving low scores if the input τ is generated by the policy network. The input of the reward network includes i) the current state $\boldsymbol{s}_t$ and action a_t, ii) the history h_{t-1} and iii) the latent vector c. The output of the network is a score from 0 to 1. Inside the reward network r, all the states, actions and the latent vector are concatenated together and pass a ConvLSTM and two fully-connected layers, the output is activated by Sigmoid function [110].

The inference network Q aims to infer the distribution of latent vector c using the generated trjectory τ. Q takes a trajectory generated by the policy network as the input, and outputs a latent vector c. Inside the inference network Q, the input trajectory passes a ConvLSTM and two fully-connected layers activated by hyperbolic tangent function [111].

4.1.3.4 Training and Testing Algorithms

To optimize Eq 5.7, novel training and testing algorithms are proposed.

STM-GAIL Training algorithm. In Eq 5.7, a prior distribution $p(c)$ is required. However, for most urban scenarios, we do not have access to $p(c)$ but instead have human

[1] In this work, we are in a model-free setup, thus, we do not need access to the transition function [108].

agent trajectories sampled from $\mathcal{T}$, we use the following generative process:

$$\tau \sim \mathcal{T}, c \sim Q(c \mid \tau) \tag{4.7}$$

to synthesize latent variables, which approximates the prior distribution when π and Q are trained to optimality, the effectiveness has been validated by Yu et al. [112]. The detailed training process is in Algorithm 9.

Input: Trajectories collected from diverse human experts $\mathcal{D} = \{\tau_i\}$, initial parameters of policy network, reward network and inference network $\boldsymbol{\theta}_0$, $\boldsymbol{\omega}_0$, $\boldsymbol{\psi}_0$.
Output: Learned policy network $\pi_{\boldsymbol{\theta}}$, reward network $r_{\boldsymbol{\omega}}$ and inference network $Q_{\boldsymbol{\psi}}$.
1: **repeat**
2: Sample two batches of trajectories τ_E and τ'_E: $\tau_E, \tau'_E \sim \mathcal{D}$
3: Infer a batch of latent codes c from τ_E: $c \sim Q_{\boldsymbol{\psi}}(c \mid \tau_E)$.
4: Sample trajectories τ using the policy network $\pi_{\boldsymbol{\theta}}$ with the latent code fixed during each rollout, *i.e.* $\tau \sim \pi_{\boldsymbol{\theta}}(\tau \mid c)$.
5: Update $\boldsymbol{\omega}$ with Adam [56] to maximize Eq. 4.8 using τ'_E and τ.
6: Update $\boldsymbol{\psi}$ with Adam [56] to minimize Eq. 4.9 using τ .
7: Update $\boldsymbol{\theta}$ with TRPO [113] to minimize Eq. 4.10.
8: **until** Convergence

Algorithm 7: STM-GAIL Training Process

Based on Eq. 5.7, we can get the objective functions for π, r and Q separately. Denote $\boldsymbol{\omega}$ as the parameters of reward network r, η as the learning rate, we update the reward network with Eq. 4.8:

$$\begin{aligned}
&\mathcal{L}_r(\boldsymbol{\omega}) = \mathbb{E}_{\pi_E}[\log(r_{\boldsymbol{\omega}}(\boldsymbol{s}, a \mid h, c))] + \mathbb{E}_{\pi}[\log(1 - r_{\boldsymbol{\omega}}(\boldsymbol{s}, a \mid h, c))], \\
&\boldsymbol{\omega} = \boldsymbol{\omega} + \eta \nabla_{\boldsymbol{\omega}} \mathcal{L}_r(\boldsymbol{\omega}).
\end{aligned} \tag{4.8}$$

Input: Trajectories $\mathcal{D}_{\text{Test}} = \{\tilde{\tau}_i\}$ collected from diverse new human experts (each expert only provides one single trajectory $\tilde{\tau}_i$), learned policy network $\pi_{\boldsymbol{\theta}}$ and inference network $Q_{\boldsymbol{\psi}}$.
Output: Generated trajectories for each human expert.
1: **repeat**
2: Infer the latent code $\tilde{c}_i$ from $\tilde{\tau}_i$: $\tilde{c}_i \sim Q_{\boldsymbol{\psi}}(c \mid \tilde{\tau}_i)$.
3: Generate trajectories $\hat{\tau}$ for the human expert using $\pi_{\boldsymbol{\theta}}$ with $\tilde{c}_i$ fixed during each rollout, *i.e.* $\hat{\tau} \sim \pi_{\boldsymbol{\theta}}(\tau \mid c_i)$.
4: **until** Testing finished for $\mathcal{D}_{\text{Test}}$

Algorithm 8: STM-GAIL Testing Process

Denote $\boldsymbol{\psi}$ as the parameters of the inference network Q, we update Q with Eq. 4.9:

$$\begin{aligned} \mathcal{L}_Q(\boldsymbol{\psi}) &= -\lambda_2 \mathbb{E}_{c\sim p(c),\tau\sim\pi(\cdot|\boldsymbol{s},h,c)}[\log Q_{\boldsymbol{\psi}}(c \mid \tau)], \\ \boldsymbol{\psi} &= \boldsymbol{\psi} - \eta \nabla_{\boldsymbol{\psi}} \mathcal{L}_Q(\boldsymbol{\psi}). \end{aligned} \tag{4.9}$$

Denote $\boldsymbol{\theta}$ as the parameters of the policy network π, our goal is to minimize the objective for $\pi_{\boldsymbol{\theta}}$ using Trusted Region Policy Optimization (TRPO) [113], the objective for $\pi_{\boldsymbol{\theta}}$ is as below:

$$\mathcal{L}_\pi(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\log(1 - r(\boldsymbol{s}, a \mid h, c))] - \lambda_1 H(\pi_{\boldsymbol{\theta}}) - \lambda_2 L_I(\pi_{\boldsymbol{\theta}}, Q). \tag{4.10}$$

STM-GAIL Testing algorithm. Given a latent vector of a new strategy, the learned policy network can easily produce more trajectories and imitate the real policy. Thus, during the testing process, we have the trajectories collected from diverse new human experts, each human expert only needs to provide one single trajectory. For each urban strategy, we first use the well-trained $Q_{\boldsymbol{\psi}}$ to infer the corresponding latent vector from a trajectory, and then use the learned $\pi_{\boldsymbol{\theta}}$ and the latent vector to produce more trajectories which are similar to the real ones governed by the real policy. The detailed testing algorithm is in Algorithm 10.

4.1.4 Evaluation

In this section, we introduce the real-world dataset, baseline models and the metrics that we use to evaluate our STM-GAIL, and present extensive experimental results.

4.1.4.1 Data and Experiment Description

In our experiment, we aim to learn the taxi drivers' passenger-seeking strategies from the collected passenger-seeking trajectories.

Dataset Description. The passenger-seeking trajectories are collected from 17,877 taxis in Shenzhen, China from July 1 to Sep 31, 2016. Each passenger-seeking trajectory is formed by multiple GPS records of a taxi. A GPS record includes five attributes including the taxi plate ID, longitude, latitude, time stamp and passenger indicator which is a binary value indicating whether a passenger is on board (*e.g.*, 0 indicates no passenger on board and 1 otherwise). The passenger-seeking trajectories are formed by consecutive GPS records with passenger indicator being 0.

To learn the passenger-seeking strategies using our STM-GAIL, we need formulate the problem in a VLMDP setup and construct the state space and action space.

State Space. We first partition the Shenzhen City into 40×50 equal-sized grid cells with a side-length $l_1 = 0.0084^\circ$ in latitude and $l_2 = 0.0126^\circ$ in longitude. And we divide the time of a day into five-minute time slots. A state of a grid cell is defined as a multi-dimensional tensor which is composed of different feature maps of its neighboring 5×5 grid cells in a specific time slot. The features include travel demand, traffic speed, taxi inflow, waiting time, distance to POIs, current location and current time slot. All the features provide the spatial and temporal information of a state, which help taxi drivers make decisions when seeking passengers. More information about the states and features is in Appendix A.

Action Space. When a taxi is in a specific state, the taxi driver has 10 actions to choose

from, including going to 8 neighboring grid cells, staying at the current grid cell, and terminating the trip.

Experiment Description. In this experiment, we study how taxi drivers make decisions when seeking passengers. Given the historical trajectories of different expert drivers in Shenzhen, China, the state space and the action space, we aim to learn the passenger-seeking strategies for diverse taxi drivers. All the expert drivers and their historical trajectories (*i.e.*, 45 expert drivers with 40 trajectories for each individual) are randomly split into training set (85%, approximately 40 drivers with 1600 trajectories) and testing set (15%, approximately 5 drivers with 200 trajectories). Our model is trained using training set and the learned policy network is tested on testing set.

4.1.4.2 Baselines

To evaluate our model, we compare STM-GAIL with state-of-the-art imitation learning methods. Firstly, to validate that the standard imitation learning methods cannot learn diverse human decision-making strategies, we compare our proposed STM-GAIL with standard GAIL and TrajGAIL listed below:

- **GAIL [86].** GAIL models human decision-making processes as MDPs, and it contains a policy network $\pi(a \mid \boldsymbol{s})$ and a reward network $r(\boldsymbol{s}, a)$. Both of the two networks are composed of several fully-connected layers.

- **TrajGAIL [89].** TrajGAIL models human decision-making processes as VLMDPs, and it contains a policy network $\pi(a \mid \boldsymbol{s}, h)$ and a reward network $r(\boldsymbol{s}, a \mid h)$. TrajGAIL applies self-attention [50] mechanism in each network to tackle the temporal dependencies of decisions.

Next, we compare our STM-GAIL with state-of-the-art meta imitation learning methods including cGAIL, InfoGAIL and MetaIL, which do not consider the spatial-temporal

dependencies in the human decision-making processes:

- **cGAIL [88].** cGAIL tries to learn strategies for different taxi drivers by incorporating the driver ID as a condition c into the policy network $\pi(a \mid \boldsymbol{s}, c)$ and reward network $r(\boldsymbol{s}, a \mid c)$. cGAIL models human decision-making processes as MDPs, and applies convolutional layers in each network.

- **InfoGAIL [90].** InfoGAIL models human decision-making processes as MDPs and aims to learn diverse strategies for agents. InfoGAIL has a policy network $\pi(a \mid \boldsymbol{s}, c)$ which incorporates a latent vector c indicating a specific task, a reward network $r(\boldsymbol{s}, a)$ shared by all tasks, and an inference network $Q(c \mid \tau)$, all the model components apply fully-connected layers inside.

- **MetaIL [91].** MetaIL is a One-Shot Imitation Learning method aiming to learn an initialized policy network which can be fast adapted to new tasks. MetaIL is trained on tasks belonging to one distribution and tested on new tasks from the same task distribution.

Moreover, to validate both spatial and temporal dependencies are important when learning diverse human strategies, we have two baseline models including Temporal Meta-GAIL (TM-GAIL) and Spatial Meta-GAIL (SM-GAIL):

- **TM-GAIL [89, 90].** Temporal Meta-GAIL (TM-GAIL) has the same objective as our STM-GAIL. However, it ignores the spatial patterns in the decision-making processes, and only applies LSTM[58] inside each network to capture the long-term dependencies of decisions.

- **SM-GAIL [89, 90].** Spatial Meta-GAIL (SM-GAIL) has the same objective as STM-GAIL. However, it ignores the long-term dependencies of decisions, and only applies convolutional layers inside each network to capture the spatial dependencies.

4.1.4.3 Evaluation Metrics

In our experiment, we use two metrics to evaluate our STM-GAIL including i) Jensen–Shannon (JS) divergence ii) L_2-Norm:

- **Jensen–Shannon (JS) divergence.** Jensen–Shannon divergence is a method of measuring the similarity between two probability distributions P and Q:

$$JSD(P||Q) = H(\frac{P+Q}{2}) - \frac{1}{2}(H(P) + H(Q)), \tag{4.11}$$

  where $H(P)$ is the Shannon entropy for distribution P. In our experiments, JS divergence is used to measure the similarity between the learned policy (*i.e.*, π) and the empirical ground-truth policy (*i.e.*, π_E).

- **L_2-Norm.** L_2-Norm (*i.e.*, Euclidean distance) is used to measure the distance between the trajectories generated by the learned policy and the trajectories sampled from the empirical ground-truth policy. L_2-Norm is defined as below:

$$L_2(P, Q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}, \tag{4.12}$$

  where $P = (p_1, \cdots, p_n)$ and $Q = (q_1, \cdots, q_n)$ can be viewed as two trajectories.

4.1.4.4 Experimental Settings

In the experiment, we parametrize the auxiliary distribution $Q(c \mid \tau)$ as a neural network, and its form depends on the true posterior $P(c \mid \tau)$. We found that simply treating $Q(c \mid \tau)$ as a factored Gaussian distribution is sufficient.

For all experiments, we use Adam[56] for online optimization. The detailed structure of STM-GAIL is as follows: the policy network π contains one ConvLSTM layer and

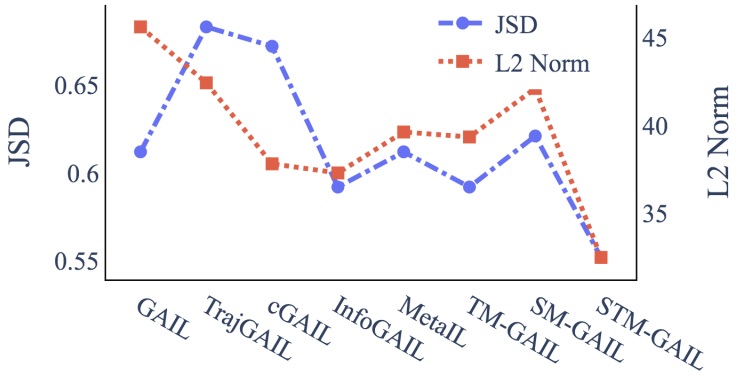
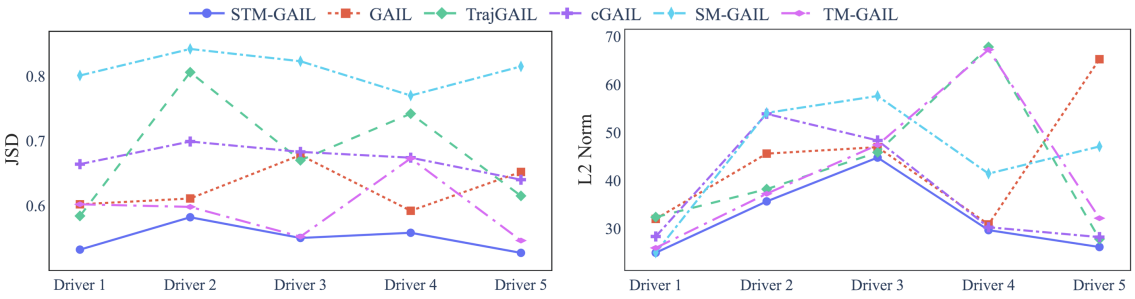


Figure 4.5: Overall performance.

one fully-connected layer activated by softmax function. The reward network r contains one ConvLSTM layer and two fully-connected layers, the final fully-connected layer is activated by Sigmoid. The Inference network Q has the same architecture as the reward network with one ConvLSTM layer and two fully-connected layers, the final layer is activated by hyperbolic tangent. The kernel size for all ConvLSTM layers is (3, 3). During training, the batch size is set to 64, and the learning rate is 1×10^{-5}.

4.1.4.5 Results

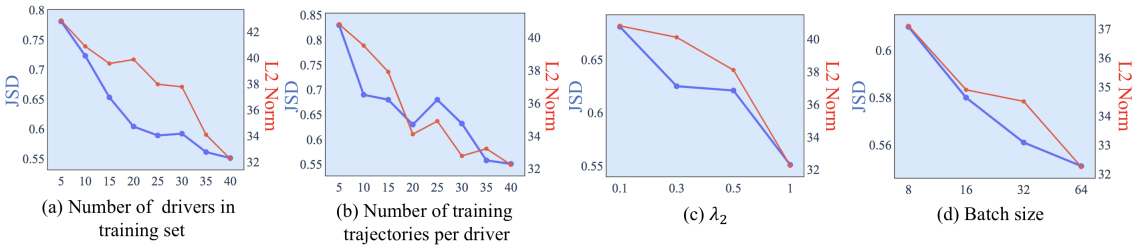
Overall performance. We first present the overall performance of our STM-GAIL compared with all baseline models when learning taxi drivers' diverse passenger-seeking strategies. For each model, we use 1600 trajectories collected from 40 taxi drivers as the training set (*i.e.*, each taxi driver provides 40 historical trajectories), and select 5 new drivers (each provides one single trajectory) as the testing set. For each new driver, we first infer his/her latent vector with the trajectory and use the learned policy to produce 100 trajectories. To evaluate whether the learned policy network with a specific latent vector is close to the taxi driver's real policy, we compare the generated trajectories with the ground-truth trajectories for each taxi driver and calculate the JS Divergence (JSD) and L_2 Norm. In this part, we test each taxi driver for three times, and present the average



(a) JS Divergence for different testing drivers

(b) L2 Norm for different testing drivers

Figure 4.6: Performance on learning diverse strategies.



(a) Number of drivers in training set

(b) Number of training trajectories per driver

(c) λ_2

(d) Batch size

Figure 4.7: Impact of hyper-parameters on urban strategies learning with STM-GAIL.

JS Divergence (JSD) and L_2-Norm of all testing drivers.

As shown in Figure 4.5, compared with our STM-GAIL, we find the imitation learning methods including GAIL and TrajGAIL have higher errors for both metrics, which indicates the two models cannot distinguish different strategies and fail to learn diverse human strategies since they assume all training trajectories are from one single expert driver. Besides, the meta imitation learning methods including cGAIL, InfoGAIL and MetaIL simply model the human decision-making processes as MDPs and ignore the spatial-temporal dependencies of human decisions, which usually leads to poor performance when learning diverse urban strategies. The higher errors of TM-GAIL and SM-GAIL indicate both spatial and temporal dependencies are important when learning diverse human urban strategies. In our proposed STM-GAIL where ConvLSTM is employed in each model component, it can successfully capture spatial-temporal dependencies of human decisions and distinguish different human expert strategies.

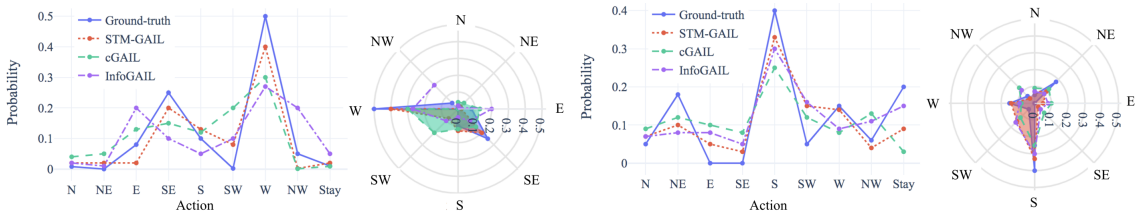
Performance on learning diverse strategies. Since our STM-GAIL is able to learn

diverse human urban strategies, we further investigate whether it can successfully mimic diverse expert drivers' behaviors. In this section, we present the statistics for each testing driver in our testing set.

As shown in Figure 4.6, we compare our STM-GAIL with some competitive baseline models. For each testing driver in our testing set, STM-GAIL presents the lowest errors for both metrics (see Figure 4.6(a) and Figure 4.6(b)). GAIL and TrajGAIL cannot distinguish taxi drivers' different strategies; cGAIL only use the driver ID to tell different drivers instead of learning the unstructured patterns and connections among strategies, which leads to higher errors and vibrations in strategies learning; SM-GAIL cannot successfully capture the temporal dependencies of human decisions and thus produces poor performance; TM-GAIL ignores the spatial patterns in the decision-making process which results in higher errors. By contrast, our STM-GAIL learns the unstructured patterns of diverse strategies using an inference network, and models the decision-making processes as VLMDPs, which guarantee the good performance in learning diverse human urban strategies.

Ablation Study. We also study how hyper-parameters influence the strategy learning performance in our STM-GAIL. In this section, the evaluated hyper-parameters include the number of drivers in training set, the number of training trajectories each driver provides, batch size and λ_2 in our objective Eq 5.7.

As shown in Figure 4.7(a), we find if the training trajectories are collected from more taxi drivers, the learned policy would be better adapted to different testing drivers' strategies. In Figure 4.7(b), we find if each driver provides more trajectories in the training process, STM-GAIL can learn diverse driving strategies better and thus produce lower errors. In Figure 4.7(c), we can find the performance is sensitive to the value of λ_2, λ_2 should be chosen based on the loss scale to ensure the whole loss scale keeps the same, in our experiments, the best choice of λ_2 should be 1. In Figure 4.7(d), we find the batch size



(a) Case 1: action selection for driver 1

(b) Case 2: action selection for driver 2

Figure 4.8: Case studies: learned policies vs. ground-truth policies for two taxi drivers in two cases.

also influences the learning performance. Large batch size results in good performance in our experiments.

Case Study. To further investigate how STM-GAIL performs when learning different urban strategies in different scenarios, we directly compare the learned policies of STM-GAIL with the empirical ground-truth policies in a few representative case studies.

We first select two taxi drivers and get their empirical policies from their mobility data. For a specific state, we present the probabilities of choosing different actions using the learned policies of STM-GAIL, cGAIL and InfoGAIL. We find that for both testing drivers in two different representative states (see Figure 5.11(a) and Figure 5.11(b)), baseline models including cGAIL and InfoGAIL do not present stable performance in action selection. By contrast, the policies learned by STM-GAIL match the ground-truth policies very well, which indicate our STM-GAIL can successfully learn diverse urban strategies by capturing spatial-temporal dependencies of decisions and learning the latent space of strategies.

5

Urban Traffic Dynamics Prediction

5.1 Continuous Spatial-Temporal Meta-Learning

5.1.1 Overview

5.1.1.1 Introduction

Over past a few decades, the rapid population growth has accelerated the process of urbanization, which in turn brings huge impacts on the urban traffic including the increasing traffic volume, worse traffic condition and the overload of the transportation infrastructures. As a result, accurately predicting the *highly dynamic traffic status* (*e.g.*, traffic volume, speed and inflow) has become a crucial work for urban development aiming to reduce congestion and increase mobility, since it can not only provide insights for urban planning, help to improve the efficiency of public transportation, but also guarantee the public safety [36].

Given the underlying road network and the historical traffic observations, *the problem of traffic dynamics prediction* aims at forecasting short-term traffic status in consecutive time slots. However, there are many practical challenges before solving this problem:

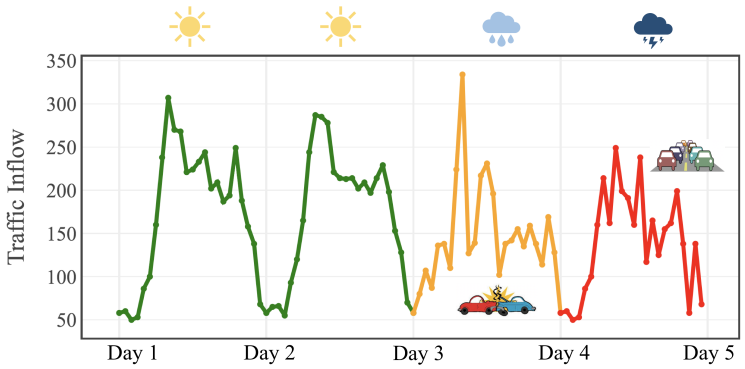


Figure 5.1: Illustration of traffic dynamics.

1) *Spatial-temporal dependencies.* It is the most common challenge when dealing with traffic dynamics prediction problem, since the traffic status would be influenced by the nearby environments, road networks and its previous traffic status.

2) *Traffic dynamics and temporal uncertainties.* In traffic dynamics prediction, the most difficult part is to capture and model the dynamics of traffic status, since urban traffic always contains temporal uncertainties due to sudden travel demand changes, unexpected events or extreme weather. For example, Figure 5.1 is an illustration of traffic dynamics, where it is possible that the traffic patterns are almost consistent in the first two days but show obvious fluctuations and temporal uncertainties in the next few days. The reasons of such considerable changes in traffic patterns could be a thunder storm, a large sports event or a car crash. In general, irregular and drastic traffic changes caused by these factors are hard to capture using traditional time series models due to their non-periodicity and rareness (*i.e.*, lacking training samples).

A lot of research efforts have been put into the traffic dynamics prediction area. Some works use traditional machine learning methods and time series models to predict urban traffic. Works such as [114, 115] and [116] use support vector regression (SVR) to capture the relationships between traffic and environmental features. Another work[117] presents a traffic prediction method which combines the SARIMA model and multi-input autore-

gressive (AR) model with genetic algorithm (GA) optimization. In addition, deep neural networks are also widely used in urban traffic prediction works. For example, works [38] and [24] predict travel demands and traffic accidents using autoencoders and ConvLSTM, respectively. Other works including [41] and [42] combine CNN and LSTM to predict the traffic speed and crowd flows. However, these models do not consider the situation where the traffic shows strong non-stationarity.

Moreover, a few works are inspired by meta-learning and try to apply existing meta-learning methods to solve the traffic dynamic prediction problem. For example, a recent work [118] combines meta graph attention and meta recurrent neural network to capture spatial and temporal dependencies simultaneously. Another work[119] learns the meta-knowledge from multiple cities and performs the spatial-temporal traffic prediction. However, these works still do not consider the temporal uncertainties in prediction and they do not extract meta-knowledge directly from traffic time series. Therefore, they have limited capabilities to learn traffic patterns that are rarely seen in the historical data.

In this paper, we try to solve the short-term traffic dynamics prediction problem and tackle the unique challenges mentioned before from the Bayesian meta-learning perspective. A novel continuous spatial-temporal meta-learner (cST-ML) is proposed, which is trained on a distribution of traffic prediction tasks generated by traffic time series data with the goal of learning a strategy that can be quickly generalized to related but unseen traffic prediction tasks from the same task distribution. cST-ML captures the spatial-temporal dependencies of traffic as well as the temporal uncertainties and dynamics through variational inference and rolling windows. Our **main contributions** are summarized as follows:

- We are the first to solve the traffic dynamics prediction problem from the Bayesian meta-learning perspective and propose a novel continuous spatial-temporal meta-learner cST-ML. cST-ML advances the Bayesian black-box meta-learning framework to cap-

ture traffic dynamics and temporal uncertainties. (See Sec 5.1.3.2.)

- cST-ML features some novel designs in the architecture. cST-ML is composed of an inference network and a decoder, where CNN and LSTM are embedded to realize the goal of capturing traffic spatial-temporal dependencies. Novel algorithms are also designed for cST-ML training and testing. During meta-training and testing, in each task, cST-ML performs traffic prediction as a rolling window which not only keeps the task uncertainties but also maintains the temporal uncertainties within each task. (See Sec 5.1.3.3 and Sec 5.1.3.4.)

- We conduct extensive experiments on two real-world traffic datasets (taxi inflow and traffic speed) to evaluate our proposed cST-ML. The experimental results verify that cST-ML can significantly improve the urban traffic prediction performance and outperform all existing baseline methods on both datasets especially when obvious traffic dynamics and temporal uncertainties are presented. (See Sec 5.1.4.)

The rest of the paper is organized as follows: Section II provides the overview of the paper, Section III details the design of cST-ML. We present the experimental results in Section IV and discuss related work in Section V. Finally, we conclude the paper in Section VI.

5.1.1.2 Preliminaries

In this section, we define the traffic dynamics prediction problem, and outline our solution framework. Table 5.1 lists the notations used in the paper.

In a city, *urban traffic status* can be characterized by many statistics, such as traffic volumes, speed, inflow/outflow, *etc*, which are of great interests to urban planners and researchers for transportation planning, traffic evaluation and more. These statistics are dynamic in nature, namely, varying across space and evolving over time. Hence, we

Table 5.1: Notations.

Notations	Descriptions
$\mathbb{S} = \{s_{ij}\}$	Grid cells
R_{ij}	A target region
$N_t \in \mathbb{N}$	Number of time slots in each task
$\tau \in \mathbb{N}$	Number of tasks
$\boldsymbol{X}^t$	Traffic related features at t
y^t	Traffic status at t
w	Rolling window size
θ	Parameters of meta-learner
$\mathcal{T}_i = \{\mathcal{D}_i^{tr}, \mathcal{D}_i^{ts}\}$	One traffic prediction task
ϕ_i	Adapted parameters for task $\mathcal{T}_i$

divide an urban area into grid cells as defined below. Each grid cell represents a target area for *urban dynamics prediction*.

Definition 1 (Grid cells). We divide a city into $I \times J$ grid cells with equal side-length (*e.g.*, $1 \times 1km$), denoted as $\mathbb{S} = \{s_{ij}\}$, where $1 \leq i \leq I, 1 \leq j \leq J$.

Definition 2 (Target region for a target grid cell). For a target grid cell s_{ij}, its target region is a square geographic region with s_{ij} in center, formed by $\ell \times \ell$ grid cells, denoted with $R_{ij} = \langle s_{ij}, \ell \rangle$.

In our study, we assume the traffic status in a target grid cell s_{ij} has high spatial correlations with the other grid cells within its target region.

Definition 3 (Traffic related features). All features that will influence the traffic status are traffic related features, *e.g.*, time of the day, travel demand, *etc*. For a grid cell s, we denote x^t as one feature of s in time slot t. For a target region R, we denote X^t as one feature map of R in time slot t, where X^t is a $\ell \times \ell$ matrix. Since there could be multiple features, all the feature maps in region R in time slot t can be denoted with a tensor $\boldsymbol{X}^t = \{X_1^t, \ldots, X_n^t\} \in \mathbb{R}^{n \times \ell \times \ell}$, where $n \in \mathbb{N}^+$ is the number of features.

Definition 4 (Traffic status). Traffic status indicates the quality of traffic, which can be measured by traffic inflow/outflow, average driving speed, *etc*. We denote y^t as the average traffic status of grid cell s in time slot t.

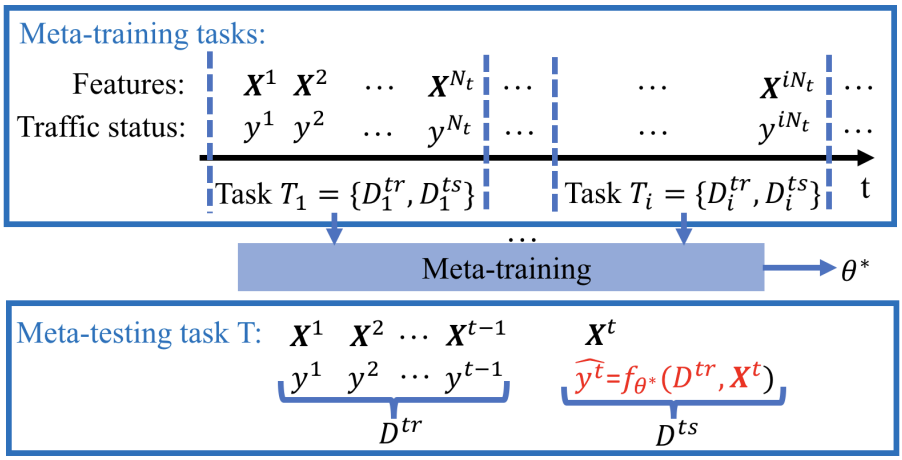


Figure 5.2: Problem illustration.

In this paper, we choose one specific measure of traffic status as the target of prediction, other measures if available can be treated as traffic related features during prediction.

Definition 5 (Traffic prediction task.) A traffic prediction task $\mathcal{T}_i$ is composed of a set of paired $(\boldsymbol{X}^t, y^t)$ in N_t consecutive time slots, which is divided into a training set $\mathcal{D}_i^{\text{tr}}$ and a testing set $\mathcal{D}_i^{\text{ts}}$, *i.e.*, $\mathcal{T}_i = \{(\boldsymbol{X}_i^1, y_i^1), \ldots, (\boldsymbol{X}_i^{N_t}, y_i^{N_t})\} = \{\mathcal{D}_i^{tr}, \mathcal{D}_i^{ts}\}$.

Problem Definition. For a specific target grid cell s, given all the historical traffic data, we aim to predict the traffic status $\{\hat{y}^t\}$ in consecutive time slots based on the available traffic related features $\{\boldsymbol{X}^t\}$. Since our goal is using meta-learning to solve this problem, the problem is transformed as follows:

in meta-learning setup, the historical time series traffic data is segmented into τ tasks, we assume all the tasks are sampled from the same distribution, $\mathcal{T}_i \sim p(\mathcal{T})$. During meta-training, we aim to train a meta-learner (with parameters θ) whose objective is to minimize the expected loss with respect to θ over all training tasks sampled from $p(\mathcal{T})$:

$$\theta^\star = \arg\min_\theta \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}\left(\phi_i, \mathcal{D}_i^{\text{ts}}\right), \text{and } \phi_i = f_\theta\left(\mathcal{D}_i^{\text{tr}}\right). \tag{5.1}$$

During meta-testing, the meta-learner is evaluated on unseen testing tasks from the same

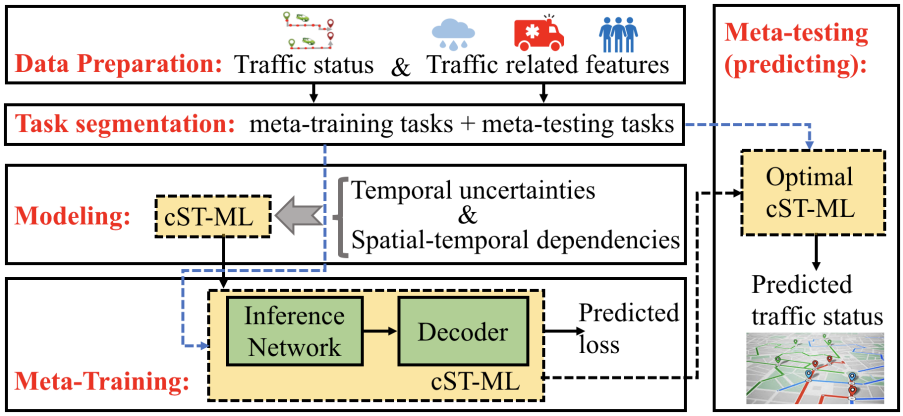


Figure 5.3: Insight of the framework.

task distribution. When predicting the future traffic, which can be view as a new testing task, we have:

$$\hat{y}^t = f_{\theta^\star}\left(\mathcal{D}^{\text{tr}}, \boldsymbol{X}^t\right), \tag{5.2}$$

where $\mathcal{D}^{\text{tr}} = \{(\boldsymbol{X}^1, y^1), \ldots, (\boldsymbol{X}^{t-1}, y^{t-1})\}$ includes a few training data in the current task. The problem is illustrated in Figure 5.2.

Solution Framework Figure 5.3 shows the solution framework. All the historical traffic data including traffic status and traffic related features is segmented into different small tasks. cST-ML is modeled based on Bayesian black-box meta-learning framework combined with novel designs which help to capture traffic uncertainties and spatial-temporal dependencies. During meta-training, the cST-ML is applied to each meta-training task to perform traffic prediction, the parameters of cST-ML are updated based on the predicted loss. The well-trained cST-ML can be fast adapted to any new traffic prediction tasks and exhibit excellent performance during meta-testing time. The detailed data preparation and task segmentation process will be presented in Section IV. We will first introduce the methodologies in the next section.

5.1.2 Related Work

Urban Traffic Prediction. In urban traffic prediction area, some works focused on traffic volume and crowd flow prediction. For example, one work [17] proposed a citywide traffic volume estimation framework which combined machine learning techniques and traffic flow theory. Another work [19] developed novel real-time framework offering accurate arrival crowd flow prediction at subway stations. In addition, a lot of works adopt and advance the existing methods to predict urban traffic. For example, works such as [114, 115] and [116] applied support vector regression to predict future traffic and took environmental features into consideration. The paper [117] proposed a traffic prediction method which combined SARIMA model and autoregressive model with genetic algorithm optimization. Another paper [45] tried to predict citywide flow using CNN which better captured traffic spatial dependencies. Yuan et al. [24] tried to predict traffic accidents with ConvLSTM. In our work, we aim to solve the urban traffic prediction problem using Bayesian meta-learning framework and capture traffic spatial-temporal dependencies and temporal uncertainties simultaneously.

Meta-Learning. A meta-learner is learned from training tasks and can be fast adapted into new tasks with just a few samples. The idea of meta-learning has been applied to many areas including supervised/unsupervised learning, reinforcement learning and even image generation. The state-of-the-art meta-learning methods including MAML [85], Reptile [82], SNAIL [120], MOCA [97], *etc*. MAMAL and Reptile learn a good initialization of a model which can be finetuned in new tasks. SNAIL is a black-box meta-learning method, where the black-box can be viewed as the meta-learner. They are not Bayesian meta-learning methods and do not consider any task uncertainties. MOCA augments a meta-learning algorithm with a differentiable Bayesian changepoint detection scheme, but it is not used to deal with time series predictions. In traffic prediction area, some works applied meta-learning methods to solve traffic prediction problem. For ex-

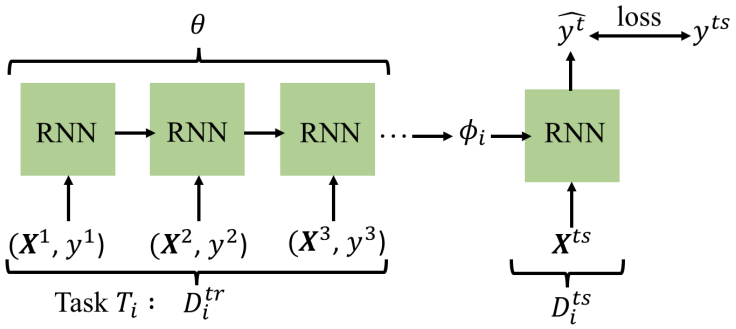


Figure 5.4: Deterministic black-box meta-learning.

ample, the work [118] combined meta graph attention and meta recurrent neural network to capture spatial and temporal dependencies simultaneously. Another work [119] learn the meta-knowledge from multiple cities and perform the spatial-temporal traffic prediction. However, these two works still did not consider the task and temporal uncertainties in prediction.

5.1.3 Methodology

In this section, we detail the key challenges of the urban dynamics prediction problem, and introduce our continuous spatial-temporal meta-learning framework.

5.1.3.1 Key challenges

State-of-the-art Meta-Learning. The goal of meta-learning is to train a model that can quickly adapt to a new task using only a few data points. To accomplish this, the meta-learner f_θ is trained during a meta-training process on a set of training tasks which are sample from the same task distribution, *i.e.*, $\mathcal{T}_i \sim p(\mathcal{T})$, such that the trained meta-learner can quickly adapt to new unseen tasks using only a small number of examples. In effect, the meta-learning problem treats entire tasks as training examples and it is a generalization across tasks rather than across data points. For each task $\mathcal{T}_i$, there are two sets of data $\mathcal{D}_i^{\text{tr}}$ and $\mathcal{D}_i^{\text{ts}}$, where $\mathcal{D}_i^{\text{tr}}$ is for task adaption and getting task specific parameters ϕ_i, $\mathcal{D}_i^{\text{ts}}$ is used

for calculating the loss and updating meta-learner parameters θ. The deterministic black-box meta-learner's objective is the same as Eq. 5.1, where the loss function $\mathcal{L}(\phi_i, \mathcal{D}_i^{\text{ts}})$ can be mean-squared error.

The deterministic black-box meta-learning framework is illustrated in Figure 5.4. The common structure of black-box meta-learning is RNN based, where for task $\mathcal{T}_i$, the parameters of RNN can be viewed as θ, the hidden state ϕ_i is the adapted parameters for the current task, and the last cell of RNN is used for testing. Thus, the distribution $q\left(\phi_i|\mathcal{D}_i^{\text{tr}}, \theta\right)$ is deterministic in this setup, which means there is no uncertainties in $\mathcal{D}_i^{\text{tr}}$, *i.e.*, $\phi_i = f_\theta\left(\mathcal{D}_i^{\text{tr}}\right)$. However, in traffic dynamics prediction problem, even though the tasks are segmented based on time (*e.g.*, everyday traffic is a task), there still exist temporal uncertainties and dynamics for each task, thus, deterministic black-box meta-learning is not enough when dealing with the traffic dynamics prediction problem.

Furthermore, Bayesian black-box meta-learning is developed to capture the task uncertainties, its objective is to maximize the log likelihood lower bound across all meta-training tasks:

$$\begin{aligned}\max_{\theta} \mathbb{E}_{\mathcal{T}_i}[\mathbb{E}_{q\left(\phi_i|\mathcal{D}_i^{\text{tr}},\theta\right)}\left[\log p\left(y_i^{\text{ts}}|\boldsymbol{X}_i^{\text{ts}}, \phi_i\right)\right] \\ - D_{KL}\left(q\left(\phi_i|\mathcal{D}_i^{\text{tr}}, \theta\right) \| p\left(\phi_i|\theta\right)\right)],\end{aligned} \tag{5.3}$$

where q is the inference network and parameterizes the mean and log-variance diagonal of a Gaussian distribution, and ϕ_i is sampled from this distribution. Since the adapted parameters ϕ_i is sampled, it can capture the uncertainties of tasks during each adaptation process.

Challenges. The Bayesian black-box meta-learning only captures the uncertainties between tasks, it does not consider the complex traffic spatial-temporal dependencies and temporal uncertainties within tasks. Thus, in traffic dynamics prediction, we need to incorporate the spatial-temporal dependencies and temporal uncertainties within tasks into the Bayesian black-box meta-learning framework and design unique structures for meta-

learner to tackle these challenges.

5.1.3.2 cST-ML Modeling

To address the challenges highlighted above, now we are in a position to develop continuous spatial-temporal meta-learning (cST-ML) framework.

Following the original Bayesian black-box meta-learning, to deal with the uncertainties in task adaptation, we treat the adapted parameters as a latent variable. We approximate the likelihood with variational lower bound (ELBO), the ELBO is derived in Eq. 5.4.

$$\begin{aligned} \log p(x) &\geq \mathbb{E}_{q(z|x)}[\log p(x,z)] + H(q(z|x)) \\ &= \mathbb{E}_{q(z|x)}[\log p(x|z)] - D_{KL}(q(z|x)\|p(z)), \end{aligned} \tag{5.4}$$

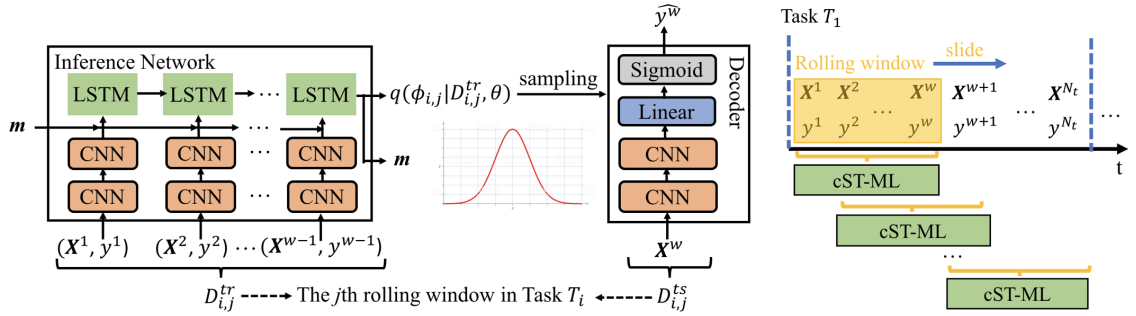
where z is the latent variable and x is the real data, D_{KL} is the Kullback-Leibler divergence, $p(x|z)$ can be treated as an decoder and $q(z|x)$ is the inference network, $p(z) \sim N(0,1)$.

In traffic dynamics prediction, to advance the Bayesian black-box meta-learning framework and take uncertainties within tasks into consideration, we first segment the historical traffic data into τ tasks, for each task, instead of directly dividing the current task into $\mathcal{D}_i^{\text{tr}}$ and $\mathcal{D}_i^{\text{ts}}$ and applying cST-ML only once, we slide cST-ML as a rolling window within the task. Just as illustrated in Figure 5.5(b). The rolling windows capture the inner temporal uncertainties within tasks and thus help to improve the prediction accuracy.

In this situation, for task $\mathcal{T}_i$ and its jth rolling window, we have specific $\mathcal{D}_{i,j}^{\text{tr}}$ and $\mathcal{D}_{i,j}^{\text{ts}}$. Eq.5.5 is the log likelihood lower bound of the jth rolling window in task $\mathcal{T}_i$:

$$\begin{aligned} L_\theta(\phi_{i,j}, \mathcal{D}_{i,j}^{\text{ts}}) = \mathbb{E}_{q(\phi_{i,j}|\mathcal{D}_{i,j}^{\text{tr}},\theta)} \left[\log p\left(y_{i,j}^{\text{ts}}|\boldsymbol{X}_{i,j}^{\text{ts}}, \phi\right)\right] \\ -D_{KL}\left(q\left(\phi_{i,j}|\mathcal{D}_{i,j}^{\text{tr}}, \theta\right)\|p(\phi_{i,j}|\theta)\right), \end{aligned} \tag{5.5}$$

where q is the inference network and parameterizes the mean and log-variance diagonal of



(a) cST-ML structure (b) Rolling window illustration

Figure 5.5: cST-ML performs as a rolling window.

a Gaussian distribution, and $\phi_{i,j}$ is sampled from this distribution for each rolling window, the Kullback-Leibler divergence can be approximated using the reparameterization trick (see more information in [121]). Compared with Eq.5.4, the latent variable corresponds to the adapted parameter $\phi_{i,j}$, and the information we use to infer $\phi_{i,j}$ includes $\mathcal{D}_{i,j}^{\text{tr}}$ and θ.

The log likelihood lower bound of all rolling windows within task $\mathcal{T}_i$ is presented in Eq.5.6.

$$L_\theta(\mathcal{T}_i) = \sum_j L_\theta(\phi_{i,j}, \mathcal{D}_{i,j}^{\text{ts}})). \tag{5.6}$$

Thus, in traffic dynamics prediction problem, the final objective is to maximize the log likelihood lower bound across all meta-training tasks:

$$\max_\theta \mathbb{E}_{\mathcal{T}_i}[L_\theta(\mathcal{T}_i)]. \tag{5.7}$$

5.1.3.3 cST-ML Architecture

We also design unique structures for cST-ML to tackle the complex spatial-temporal traffic dependencies. The structure of our cST-ML is composed of an inference network and a decoder. The inference network tries to encode the training data within a task into a latent distribution which captures the spatial patterns of the current location and also learns

the temporal dependencies and uncertainties, the decoder is responsible for the prediction using the testing data within the same task. Figure 5.5 shows the overall structure of cST-ML.

The Inference Network is CNN and LSTM based and is actually the adaptation process of a task, which takes in the $\mathcal{D}_{i,j}^{\text{tr}}$ and extracts information from $\mathcal{D}_{i,j}^{\text{tr}}$, aiming to output a latent distribution which captures uncertainties of the jth rolling window in $\mathcal{T}_i$. The input of the inference network includes two parts, i) $\mathcal{D}_{i,j}^{\text{tr}} = \{(\boldsymbol{X}_i^1, y_i^1), \ldots, (\boldsymbol{X}_i^t, y_i^t)\}$, where $t < N_t$, and ii) a vector $\boldsymbol{m}$ containing memories from the previous rolling windows within the current task $\mathcal{T}_i$, which is actually the hidden state of the LSTM in the last time step. In the first rolling window of $\mathcal{T}_i$, the input memory is a zero vector.

Since $\boldsymbol{X}^t$ is a tensor for each time slot, y^t is first enlarged to an $\ell \times \ell$ matrix and concatenates with $\boldsymbol{X}^t$, and then the concatenated tensor goes through a few layers of CNN activated by ReLU which can capture the spatial dependencies of local traffic. The output sequence then concatenates with the memory vector and becomes the input of the LSTM, the hidden state of LSTM in the last time slot t goes through fully-connected layers and produces the mean and log variance of a Gaussian distribution $q\left(\phi_{i,j}|\mathcal{D}_{i,j}^{\text{tr}}, \theta\right)$.

The Decoder aims to produce the prediction $\hat{y}^{ts}$ based on $\boldsymbol{X}^{ts}$ where $(\boldsymbol{X}^{ts}, y^{ts}) \in \mathcal{D}_{i,j}^{\text{ts}}$, the prediction loss is calculated using y^{ts} and $\hat{y}^{ts}$. Decoder takes two inputs, i) the adapted information $\phi_{i,j}$ sampled from $q\left(\phi_{i,j}|\mathcal{D}_{i,j}^{\text{tr}}, \theta\right)$ and ii) $\boldsymbol{X}^{ts}$. $\boldsymbol{X}^{ts}$ first goes through a few layers of CNN activated by ReLU and then concatenates with $\phi_{i,j}$, the results pass fully-connected layers activated by Sigmoid function and we get the final prediction $\hat{y}^{ts}$. The detailed structure of the inference network and decoder are illustrated in Figure 5.5(a).

Input: Task distribution $p(\mathcal{T})$, window size w, step size $c = 1$, initialized cST-ML f_{θ_0}.
Output: Well trained cST-ML.

while not done **do**
  Sample a task $\mathcal{T}_i \sim p(\mathcal{T})$.
  Prepare $\mathcal{D}_{i,j}^{\text{tr}}$ and $\mathcal{D}_{i,j}^{\text{ts}}$ for each rolling window in $\mathcal{T}_i$
  for all rolling windows in $\mathcal{T}_i$ **do**
    Sample $\phi_{i,j}$ from $q\left(\phi_{i,j}|\mathcal{D}_{i,j}^{\text{tr}}, \theta\right)$.
    Compute log likelihood using Eq.5.5.
  end for
  Update θ with Adam [56] to maximize Eq.5.6.
end while

Algorithm 9: Meta-Training

5.1.3.4 cST-ML Training and Testing

Since we perform cST-ML as a rolling window within tasks, assume the rolling window size is w and step size is 1, for the 1st rolling window, we use the first $w-1$ data points $\{(\boldsymbol{X}_i^1, y_i^1), \ldots, (\boldsymbol{X}_i^{w-1}, y_i^{w-1})\}$ in $\mathcal{T}_i$ as input of the inference network and use $\boldsymbol{X}_i^w$ as the input of decoder to predict $\hat{y}_i^w$, thus, $\mathcal{D}_{i,j}^{\text{tr}} = \{(\boldsymbol{X}_i^1, y_i^1), \ldots, (\boldsymbol{X}_i^{w-1}, y_i^{w-1})\}$ and $\mathcal{D}_{i,j}^{\text{ts}} = \{(\boldsymbol{X}_i^w, y_i^w)\}$, just as illustrated in Figure 5.5. Then, we use data points $\{(\boldsymbol{X}_i^2, y_i^2), \ldots, (\boldsymbol{X}_i^w, y_i^w)\}$ as input of inference network and use $\boldsymbol{X}^{w+1}$ as the input of decoder to predict $\hat{y}^{w+1}$ and so on so forth. In this situation, for task $\mathcal{T}_i$ and its jth rolling window, we have specific $\mathcal{D}_{i,j}^{\text{tr}}$ and $\mathcal{D}_{i,j}^{\text{ts}}$, and we backpropagate through the total loss of all rolling windows in task $\mathcal{T}_i$ to update meta-learner θ (*i.e.*, the parameters of both inference network and decoder).

The detailed meta-training process is shown in Algorithm 9. We repeatedly sample tasks from the task distribution, for each sampled task, we compute the total log likelihood for all rolling windows and update θ once.

After training, the well-trained meta-learner θ can fast adapt to any new tasks. The meta-testing algorithm is shown in Algorithm 10. In meta-testing, to predict the future traffic, we define a new testing task $\mathcal{T} = \{(\boldsymbol{X}^1, y^1), \ldots, (\boldsymbol{X}^{t-1}, y^{t-1})\}$, after the predic-

Input: A new task $\mathcal{T} = \{(\boldsymbol{X}^1, y^1), \ldots, (\boldsymbol{X}^{t-1}, y^{t-1})\}$ with available $\boldsymbol{X}^t, \ldots, \boldsymbol{X}^{N_t}$, window size $w = t$, step size $c = 1$, well-trained cST-ML $f_{\theta^\star}$.
Output: Predicted values $\{\hat{y}^t, \ldots, \hat{y}^{N_t}\}$.
1: Define $\mathcal{D}^{\text{tr}} = \{(\boldsymbol{X}^1, y^1), \ldots, (\boldsymbol{X}^{t-1}, y^{t-1})\}$ and $\mathcal{D}^{\text{ts}} = \{\boldsymbol{X}^t\}$ as the first rolling window in $\mathcal{T}$
2: **for** all rolling windows in $\mathcal{T}$ **do**
3: $\quad \hat{y}^t = f_{\theta^\star}(\mathcal{D}^{\text{tr}}, \mathcal{D}^{\text{ts}})$.
4: $\quad$ Update $\mathcal{D}^{\text{tr}} = \{(\boldsymbol{X}^2, y^2), \ldots, (\boldsymbol{X}^t, \hat{y}^t)\}$ and $\mathcal{D}^{\text{ts}} = \{\boldsymbol{X}^{t+1}\}$ for the next rolling window.
5: **end for**

Algorithm 10: Meta-Testing

tion of $\hat{y}^t$, we update the $\mathcal{D}^{\text{tr}}$ and $\mathcal{D}^{\text{ts}}$ of the current rolling window and slide the window to get more predictions.

5.1.4 Evaluation

In this section, we conduct extensive experiments on real-world traffic datasets to evaluate our cST-ML. We first describe the datasets and introduce experiments, then we present baselines compared with our model and the evaluation metrics. Finally, the experiment results are presented and analyzed in detail.

5.1.4.1 Dataset Descriptions

Preprocessing of Dataset

We evaluate our model on the real-world datasets including (1) traffic speed, (2) taxi inflow and (3) travel demand, all of which are extracted from Shenzhen, China from Jul 1st to Dec 31st. In the preprocessing step, we first apply map gridding to the whole Shenzhen City, where the city is partitioned into 40×50 grid cells, for each target grid cell, its target region is the 5×5 matrix with the target grid cell in center. Thus, there are in total $1,656$ possible target grid cells. The map gridding method, the target grid cells and its corresponding target regions are illustrated in Figure 5.6.

(a) Road map gridding (b) Target grid cells and regions

Figure 5.6: Map gridding and target grid cells illustration.

Traffic speed, taxi inflow and travel demand are all extracted from taxi GPS records collected in Shenzhen, China from Jul 1st to Dec 31st, 2016. In each time slot (*i.e.*, one hour) of each day, taxi inflow is the number of taxis that stay or arrive at a target grid cell, travel demand is the number of taxi pickups within a target grid cell. In effect, it is hard to obtain the travel demands of all transport modes in a target grid cell, thus, we use taxi demands to represent travel demands, and many studies have shown that taxi demand is a very representative measure of travel demand [14, 15].

Experiment Descriptions

Next, we describe our two traffic prediction experiments we will perform in detail.

- **Traffic speed prediction.** In speed prediction, the traffic status in each grid cell is measured by average traffic speed, and there are 12 time slots per task, *i.e.*, $N_t = 12$, and thus 184 tasks over 6 months. All the tasks are divided into meta-training tasks (the first 80% of all tasks) and meta-testing tasks (the rest of 20%). We treat travel demands, traffic inflow and the time of the day as traffic related features, and use meta-training tasks to train the model and use meta-testing tasks to do evaluations. The goal of this task is to predict the traffic speed of a target grid cell s based on the historical available features.

- **Taxi inflow prediction.** Similar to traffic speed prediction, in the taxi inflow prediction, the traffic status in each grid cell is measured by taxi inflow. We view travel demands,

traffic speed and the time of the day as traffic related features. There are also 12 time slots per task, *i.e.*, $N_t = 12$, and 184 tasks over 6 months. All the tasks are divided into meta-training tasks (the first 80% of all tasks) and meta-testing tasks (the rest of 20%). We aim to train the model with meta-training tasks and evaluate the model using meta-testing tasks.

5.1.4.2 Baselines

- **HA [119]** For each grid cell, Historical Average method (HA) predicts the traffic status for a target grid cell based on its average status of the previous time slots.
- **Regression [35].** This method applies ridge regression to predict the future traffic status, the predictors are the corresponding traffic related features. The training data are used to train the regression model and the testing data are used for evaluations.
- **ARIMA [122].** Auto-Regressive Integrated Moving Average (ARIMA) is a conventional parametric based time-series model. Here we view the historical traffic status as time-series data and apply ARIMA to predict the future traffic status.
- **LSTM [52, 54].** This method uses LSTM to predict the future traffic status using traffic related features as input. The daily traffic related features can be viewed as an input sequence, which goes through CNNs first and then passes LSTM to get the predicted traffic status sequence.
- **SNAIL [120].** It is an state-of-the-art deterministic black-box meta-learning method. SNAIL utilizes attention layers to get the deterministic adapted parameters for each task instead of sampling from a distribution, where the task uncertainties are not considered.
- **BBML [123].** It is the Bayesian black-box meta learning method without memory vector and rolling windows. The structure of this baseline is same as cST-ML, however,

it does not apply rolling windows and there is no memories kept within a task.

5.1.4.3 Evaluation Metrics

We use mean absolute percentage error (MAPE) and rooted mean square error (RMSE) for evaluations:

$$\text{MAPE} = \frac{1}{T}\sum_{t=1}^{T} |y_t - \hat{y}_t| / y_t, \tag{5.8}$$

$$\text{RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T} (y_t - \hat{y}_t)^2}, \tag{5.9}$$

where y_t is the ground-truth traffic status observed in the target grid cell s in the t-th time slot, and $\hat{y}_t$ is the corresponding prediction, T is the total number of time slots to perform prediction.

5.1.4.4 Experimental Settings

The whole Shenzhen city is divided into 40×50 grid cells with a side-length $l_1 = 0.0084°$ in latitude and $l_2 = 0.0126°$ in longitude. The target region for a target grid cell is of size 5×5, *i.e.*, $\ell = 5$. Thus, there are in total $1,656$ possible target grid cells in Shenzhen city. In the experiment, we can select any possible target grid cell to perform traffic predictions.

The time interval for each task used to train the cST-ML are from 7:00am to 7:00pm, where each hour is a time slot and we have 12 time slots per day/task, *i.e.*, $N_t = 12$. Thus, we have $\mathcal{T}_i = \{(\boldsymbol{X}_i^1, y_i^1), \ldots, (\boldsymbol{X}_i^{12}, y_i^{12})\}$.

The structure of cST-ML is as follows: two layers of CNN are utilized before LSTM in the inference network, the input channel of the first CNN is 4, the output channel is 64, the kernel size is 3, stride is 1 and padding is 1; for the second CNN layer, the input channel is 64, the output channel is 128, the kernel size is 5, stride is 1 and padding is 0. In decoder, we still use two layers of CNN combined with a linear transformation. cST-ML

Table 5.2: Performance on traffic speed prediction and taxi inflow prediction.

Methods			HA	Regression	ARIMA	LSTM	SNAIL	BBML	**cST-ML**
Traffic speed	1h	RMSE	2.993	2.224	2.160	2.923	2.216	6.805	**1.055**
		MAPE	0.170	0.124	0.124	0.156	0.126	0.379	**0.058**
	3h	RMSE	2.545	2.249	2.517	2.674	2.278	5.408	**2.119**
		MAPE	0.126	0.112	0.110	0.125	0.114	0.304	**0.093**
	6h	RMSE	3.120	3.035	3.711	3.000	2.933	5.360	**2.685**
		MAPE	0.197	0.183	0.221	0.199	0.171	0.328	**0.164**
Taxi inflow	1h	RMSE	56.833	37.356	26.902	37.501	29.715	24.749	**16.461**
		MAPE	0.239	0.159	0.120	0.160	0.116	0.104	**0.061**
	3h	RMSE	65.929	38.572	35.551	38.940	31.191	31.382	**19.258**
		MAPE	0.237	0.145	0.119	0.147	0.097	0.117	**0.064**
	6h	RMSE	64.777	31.937	38.734	32.781	25.592	29.253	**18.744**
		MAPE	0.235	0.111	0.124	0.114	0.079	0.106	**0.061**

is trained using Adam optimizer [56] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and a learning rate of 2×10^{-4} for 2000 times task samplings, the window size is 5 with step size equal to 1.

5.1.4.5 Evaluation Results

Average prediction performance. First, we conduct experiments to compare the average prediction performance of our proposed cST-ML and six competing baseline models. The results are shown in Table 5.2. In the table, for a specific target grid cell, we present the RMSE and MAPE results for one-hour, three-hour and six-hour traffic speed prediction and taxi inflow prediction. For meta-based models (including cST-ML, BBML and SNAIL), we randomly pick 5 meta-testing tasks in both of the traffic speed and taxi inflow predictions, compute the one-hour, three-hour and six-hour RMSE and MAPE for each testing task and report the average results in the table. For other models, we use the same testing data to compute the statistics.

In traffic speed prediction, according to the average RMSE and MAPE in one-hour, three-hour and six-hour predictions, cST-ML outperforms all the baseline models. Compared with BBML model, cST-ML always has great improvements in both metrics, even though the two methods are both Bayesian black-box meta-learning based. The reason is that in cST-ML, the memory vector can help to capture the temporal dependencies of traffic within each task, and the rolling windows can better capture the temporal uncertainties

which lead to better prediction performance.

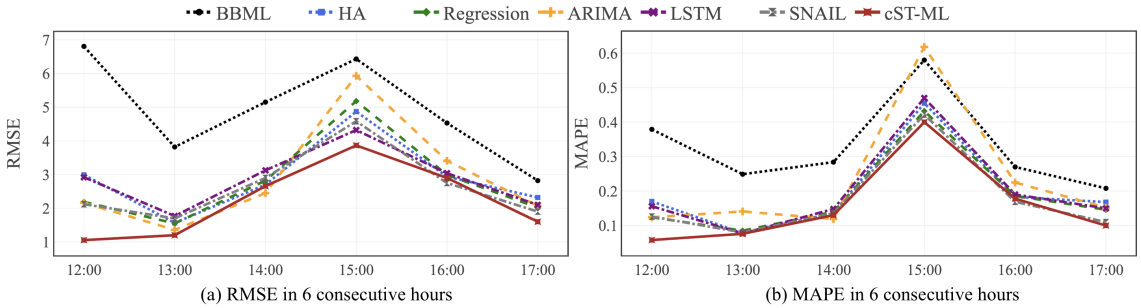
SNAIL is a deterministic black-box meta-learning method which does not consider any uncertainties of tasks, but it achieves competitive performance in some cases (*i.e.*, six-hour traffic speed and taxi inflow predictions), the reason is that we view daily traffic as one task in meta-training and meta-testing, for one specific target grid cell, in most of cases, the everyday traffic is similar which means there is less task uncertainties, and thus SNAIL can achieve competing prediction performance sometimes.

LSTM is used as a seq2seq model in traffic prediction, which utilizes the traffic related features to predict the traffic status, so it does not rely on the previous traffic status in testing or prediction process which could result in larger prediction errors.

Compared with the traditional traffic prediction baseline models including HA, Regression and ARIMA, cST-ML achieves significant improvements since it not only captures the traffic spatial-temporal dependencies but also the temporal uncertainties. On the contrary, these traditional models only consider either the temporal dependencies or the relationships between traffic status and features, and they cannot deal with the traffic uncertainties very well.

In taxi inflow prediction, we get similar prediction results. SNAIL and BBML are the most competitive baselines compared with other baseline models, which indicates they can better learns the spatial-temporal patterns of traffic, and thus obtain lower errors. However, cST-ML is more powerful due to its novel design.

Detailed performance in consecutive time slots. In this part, we are aiming to prove the effectiveness of our cST-ML in traffic predictions in each time slot (*e.g.*, one hour). In urban traffic prediction problem, the good average prediction performance is not enough, since we expect to get more accurate prediction for each specific time slot. Thus, we conduct experiments and provide detailed prediction performance for each time slot (*i.e.*, one hour). The statistics are calculated based on 5 meta-testing tasks in both of the traffic



(a) RMSE in 6 consecutive hours (b) MAPE in 6 consecutive hours

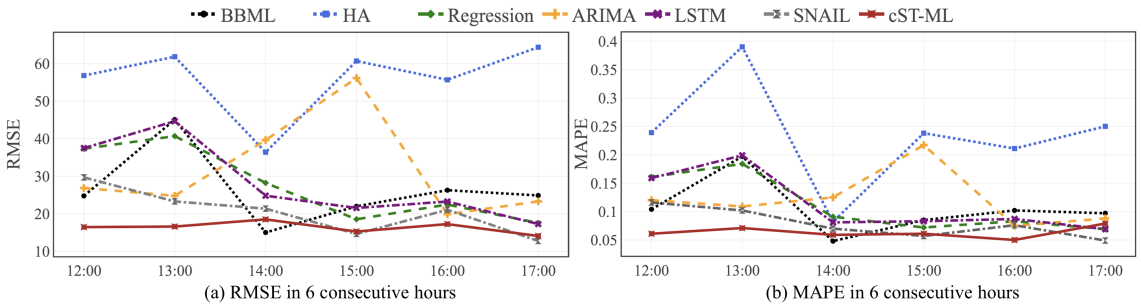
Figure 5.7: Comparisons of models in 6 consecutive hours in traffic speed prediction.

speed and taxi inflow predictions, in each time slot, we report the average RMSE and MAPE of all 5 testing tasks.

In traffic speed prediction, the detailed performance is presented in Figure 5.7. As shown in Figure 5.7(a) and Figure 5.7(b), our cST-ML achieves the best prediction performance in most of the time slots, but in some cases, some baseline models would have slightly better performance, for example, ARIMA has the best performance at 14:00 and SNAIL has the best performance at 16:00. However, the performance of baselines including ARIMA and SNAIL presents higher volatilities and thus the prediction performance is much more unstable.

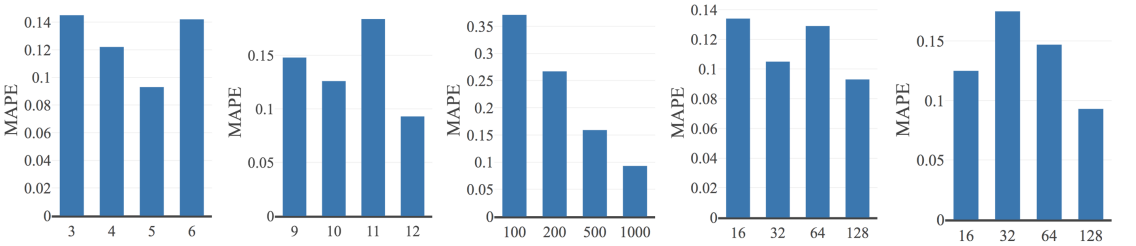
In taxi inflow prediction, as shown in Figure 5.8(a) and Figure 5.8(b), our cST-ML also achieves the best prediction performance except in a few time slots, for example, BBML and SNAIL have slightly better performance than cST-ML at 14:00 hour and 17:00, respectively. However, similar to Figure 5.7, the performance of all baseline models still presents much higher volatilities in prediction performance, in contrast, cST-ML displays more stable and accurate predictions in general, which also proves that cST-ML can better capture the traffic uncertainties and complex spatial-temporal dependencies, therefore, cST-ML provides more accurate and stable traffic prediction in consecutive time slots.

Evaluations on cST-ML parameters. cST-ML has many hyper-parameters, *e.g.*, rolling window size, task length, *etc*. In this part, we conduct experiments to evaluate the impacts



(a) RMSE in 6 consecutive hours (b) MAPE in 6 consecutive hours

Figure 5.8: Comparisons of models in 6 consecutive hours in taxi inflow prediction.

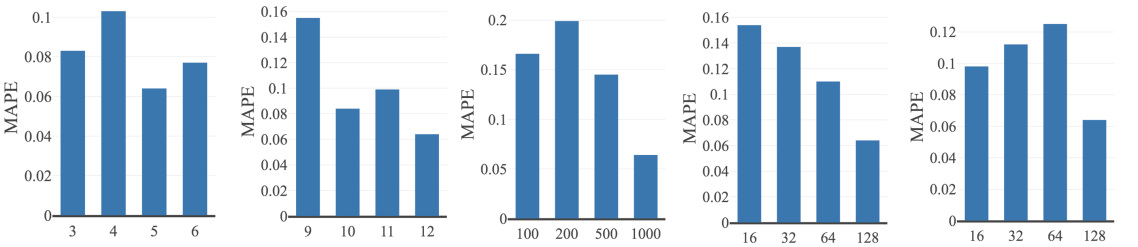


(a) Rolling window size (b) Task length (c) Number of task iterations in training (d) Dimension of hidden states (e) Dimension of μ and σ

Figure 5.9: Impact of parameters in traffic speed prediction.

of different hyper-parameters on our cST-ML. In Figure 5.9 and Figure 5.10, the experimental results are presented to demonstrate how different values of hyper-parameters influence the performance of cST-ML. Specifically, the hyper-parameters we aim to analyze includes rolling window size, task length, the number of training iterations, the dimension of hidden states in LSTM (inside the inference network of cST-ML) and the dimension of mean and log variance, which are used to define the output distribution of inference network in cST-ML. All the statistics in Figure 5.9 and Figure 5.10 are MAPEs of 3-hour predictions, which are computed using 5 meta-testing tasks in both of traffic speed prediction and taxi inflow prediction.

As shown in Figure 5.9(a) and Figure 5.10(a), the prediction performance is sensitive to rolling window size. The errors are both high when the rolling window size is too small or large. When the rolling window size is small (*e.g.*, window size equal to 3 or 4), we only use a few data points (*e.g.*, 2 or 3 data points) within a task to do the next step



(a) Rolling window size (b) Task length (c) Number of task iterations in training (d) Dimension of hidden states (e) Dimension of μ and σ

Figure 5.10: Impact of parameters in taxi inflow prediction.

prediction, where little traffic information is provided in task adaptation and thus leads to high prediction errors. However, if the the rolling window size is too large, the temporal uncertainties are less captured, when the rolling window size equal to the task length, no temporal uncertainties within a task can be captured, which also lead to poor prediction performance.

In Figure 5.9(b) and Figure 5.10(b), task length also influences the performance of our model and when task length is equal to 12, we have the best performance. Since the historical traffic data (from 7:00am to 7:00pm every day) can be viewed as time series data, if we segment the tasks by 12 hours, each task contains complete traffic information of one day. In general, everyday traffic patterns are similar, the assumption that all tasks are sampled from the same distribution is satisfied in meta-learning framework, which leads to higher prediction accuracy. On the contrary, if the tasks are not segmented by 12 hours, tasks may display greatly different traffic patterns where the meta-learning assumption is hard to be satisfied and thus usually results in high prediction errors.

In Figure 5.9(c) and Figure 5.10(c), we can easily conclude that the more training iterations we have, the lower prediction errors cST-ML produces. Since in each training iteration, we randomly sample a task from the task distribution, the more times we sample, the better meta-knowledge we get through the whole training process which certainly will lead to better performance.

We also change the dimensions of hidden states in LSTM (inside the inference net-

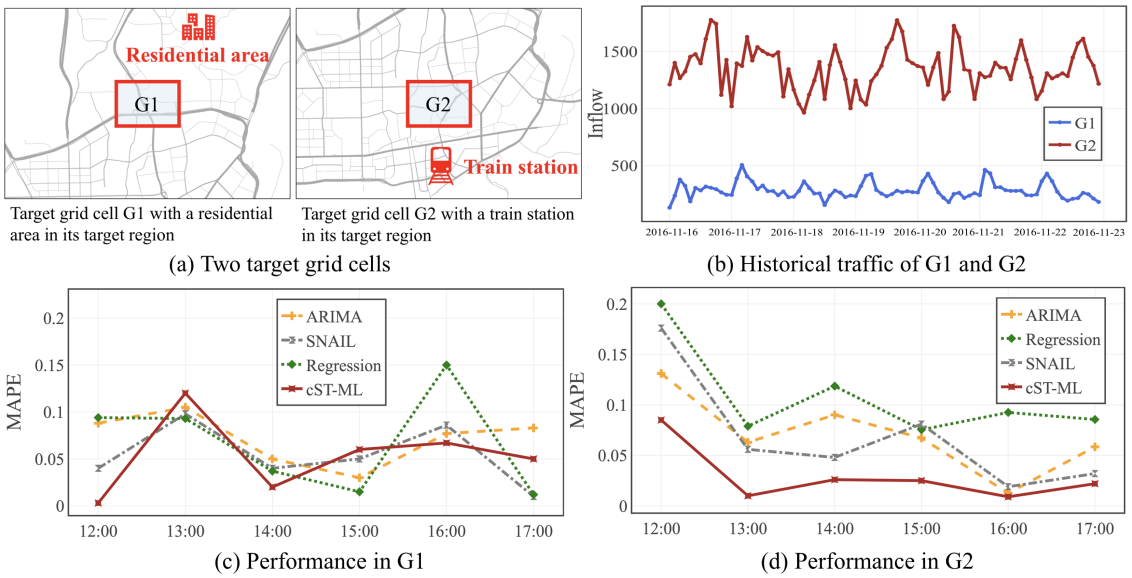


Figure 5.11: Traffic predictions of two target grid cells.

work of cST-ML). The results are shown in Figure 5.9(d) and Figure 5.10(d). We find that the model performance is very sensitive to the dimension of hidden states, higher dimension leads to better performance, which indicates more spatial and temporal information is kept by LSTM.

Similarly, we analyze how the dimension of mean and log variance influence performance in Figure 5.9(e) and Figure 5.10(e). Mean and log variance are the outputs of inference network in cST-ML, by which the distribution of adapted parameters is determined. We find that higher dimension leads to lower error, which indicates more information of the output distribution is captured in a task.

5.1.4.6 Case Studies

To further illustrate the effectiveness of our cST-ML to capture traffic dynamics, we perform a case study in this subsection. Since different locations could show different traffic patterns, *e.g.*, in Figure 5.11(a), two representative target grid cells G1 and G2 are presented, the target region of G1 contains a residential area and the target grid cell G2 is

close to a train station. As shown in Figure 5.11(b), the historical traffic (*i.e.*, 7 days taxi inflow) are plotted, where the taxi inflow of G2 presents obvious fluctuations and no regular patterns can be captured since the travel demand in G2 varies every day, while the daily traffic patterns of G1 are more consistent due to the similar daily travel demands in this area. In this case study, we study the traffic prediction performance of our cST-ML when dealing with such two different target grid cells. We compare our cST-ML with three competitive baselines and the performance is shown in Figure 5.11(c) and (d). When performing traffic prediction in a location with obvious temporal dynamics such as G2, cST-ML outperforms all other baselines, while in a location with consistent traffic patterns like G1, some baselines can provide reasonable predictions as well. This case demonstrates that our cST-ML has excellent capability to deal with traffic dynamics and temporal uncertainties and thus tends to present better performance when local traffic presents greater fluctuations and irregular patterns.

6

Conclusion and Future Work

6.1 Conclusion

In this dissertation, we will conclude all the works in the following four domains.

Conditional Urban Traffic Estimation with Generative Adversarial Networks. In this domain, we propose and investigate a novel conditional traffic estimation problem, namely, estimating the impact of travel demands on regional traffic status. Solving this problem is crucial to potentially avoid traffic issues caused by sudden greatly improved travel demands, *e.g.*, emergencies and new urban constructions. In this topic, a novel generative model - TrafficGAN was proposed. Using traffic data (*e.g.*, taxi inflow) from all regions under different travel demands, TrafficGAN is trained to capture the fundamental patterns of how traffic condition evolves with respect to the travel demand changes and underlying road network structures. With such knowledge, the obtained generator is capable of generating realistic traffic conditions within a region for a not-yet-observed travel demand. Moreover, we also propose a novel Curb-GAN to estimate urban traffic based on various travel demands. Curb-GAN is capable of modeling both spatial and temporal auto-correlations and producing a sequence of estimated traffic in consecutive time

slots. Using real-world spatio-temporal traffic data, we evaluated our Curb-GAN on two datasets. The well-trained generator is capable of generating realistic traffic distribution sequences in a region given a not-yet-observed travel demand sequence, and our proposed Curb-GAN significantly outperforms all baseline models. Besides, we propose a novel Complex-Condition-Controlled Generative Adversarial Network (C^3-GAN) to estimate the regional urban traffic based on complex urban conditions, *e.g.*, new bus routes, rainfall intensity and travel demands. In C^3-GAN, we design *i)* an embedding network to map the complex urban conditions to a latent space and extract high-quality representations of conditions, *ii)* an inference network to enhance the relations between the embedded latent vectors and the traffic data, *iii)* a unique architecture and a training algorithm to guarantee the network stability and performance. Our experimental results using real-world datasets demonstrate that our models outperforms state-of-the-art baselines in traffic estimation problems.

Transferable Generative Adversarial Networks. In this domain, we tackle the cross-city conditional traffic estimation problem in case of data scarcity, and we propose to perform traffic estimation with a novel spatial transfer generative learning framework — STrans-GAN, which combines generative models with transfer learning in multiple source cities setup. STrans-GAN preserves various traffic patterns through clustering, and incorporate meta-learning idea into the pre-training process to learn a good global generalized model. During fine-tuning, we propose to add a cluster matching regularizer aiming to realize the flexible adaptation in different scenarios. Besides, novel pre-training and fine-tuning algorithms are proposed. Through extensive experiments on multiple-city datasets, the effectiveness of STrans-GAN is proved, which significantly improves the estimation performance and outperforms all state-of-the-art baseline methods.

Learning Human Driving Strategies. In this domain, we make the first attempt to solve the human urban strategy analysis problem in case of data scarcity and data

heterogeneity, and propose a novel imitation leaning paradigm —Spatial-Temporal Meta-GAIL (STM-GAIL), which can successfully learn diverse human urban strategies from heterogeneous human-generated spatial-temporal urban data. In our STM-GAIL, we incorporate the spatial-temporal dependencies of human decisions into GAIL framework, and propose to learn diverse human urban strategies from the meta-learning perspective, where an inference network is designed on top of the standard GAIL to infer the latent variables of diverse human strategies in an unsupervised way by maximizing the mutual information between the latent space and trajectories. STM-GAIL can be generalized to a new human urban strategy with a single trajectory. Extensive experiments on real-world dataset are performed to validate the effectiveness of STM-GAIL. The experimental results show that our STM-GAIL has significant improvement compared to state-of-the-art baselines when learning human decision-making strategies.

Urban Traffic Dynamics Prediction. In this domain, we solve the traffic dynamics prediction problem using Bayesian meta-learning framework. We propose a novel continuous spatial-temporal meta-learner (cST-ML), which learns a general traffic dynamics prediction strategy from historical traffic data (segmented into tasks) and could be quickly adapted to new prediction tasks containing just a few samples and exhibited excellent prediction performance. cST-ML captures the traffic spatial-temporal dependencies and the traffic uncertainties through new features in both objective and architecture beyond the original Bayesian black-box meta-learning. Novel training and testing algorithms are designed for cST-ML where the traffic temporal uncertainties and dynamics are better kept by rolling windows. We conduct experiments on real-world traffic datasets (taxi inflow and traffic speed) to evaluate our proposed cST-ML. The experiment results verify that cST-ML can significantly improve the urban traffic prediction performance especially when obvious traffic uncertainties are presented and significantly outperforms all baseline models.

6.2 Future Work

6.2.1 Preference Reveal for Human Agents via Generative Adversarial Meta Learning.

In urban area, it is necessary to learn how the taxi drivers make decisions when seeking passengers, how they learn to drive more efficiently, and how they respond to various traffic conditions, etc. Such knowledge facilitates the understanding of the human learning processes and supports behavioral studies. In the third topic of this dissertation, with the help of large-scale urban taxi trajectory data, we investigate the behaviors and strategies of taxi drivers by designing a novel spatial-temporal meta GAIL on top of meta-learning and generative adversarial imitation learning (GAIL) to learn diverse human driver strategies and thus improve the business practices of taxi services.

In the future, I will also look into the problems of learning preferences from more diverse human agents. For example, for financial agents in financial area, current methods of extracting human financial agents' investing preference is Robo-advising techniques, where the robo advisors are expected to provide individual-specific financial advice to its clients based on their own preference. Recovering an agent's preference via some traditional methods like questionnaire, however, may lead to strong cognitive and behavioral bias. Learning from demonstrations methods such as inverse reinforcement learning and imitation learning can help to solve the preference reveal problem by learning the investment preference directly from people's past investment behaviors. In the future, I will focus on developing new techniques to recover the preference of various human agents from their past behaviors.

6.2.2 Explainable Generative Adversarial Networks

In this dissertation, we mainly study the generative adversarial network and its diverse applications on spatial-temporal urban big data, however, all of the generative models are ”black-box” in nature, which means it is hard to explain the connection between the input and output, and it is also difficult to interpret how each layer inside the generator or discriminator influences the final model performance. In my future research, we can focus more on developing explainable generative adversarial networks for Spatial-temporal Urban Data. Explainable generative adversarial networks is a sub-area of Explainable Artificial Intelligence (XAI), which can help to explain the function of neural networks in side GANs and interpret the model output. For instance, the STrans-GAN introduced in the second topic of this dissertation can be further explained to help urban decision-makers understand what kind of cities are more appropriate to be treated as source cities for a specific target city.

6.2.3 Novel Application Domains in Urban Intelligence

In this dissertation, we mainly study the problems in the domains of urban traffic estimation and human behavior analysis. There are still many other application problems which would benefit a lot from current techniques of spatial-temporal urban big data analytics, such as urban accidents prediction problem (e.g., crime accidents prediction and traffic accidents prediction), precipitation nowcasting problem, human mobility analysis, smart taxis problems, etc. These problems are also highly related to spatial-temporal data, and in the future, it is possible to capture both spatial and temporal dependencies within the data with novel graph neural networks, attention-based models, etc.

References

[1] Yingxue Zhang, Yanhua Li, Xun Zhou, Xiangnan Kong, and Jun Luo. TrafficGAN: Off-deployment traffic estimation with traffic generative adversarial networks. In *ICDM*, 2019. 5, 34, 35, 37, 40, 48, 58, 61, 76, 83, 87, 88, 90, 99, 101, 110

[2] Yingxue Zhang, Yanhua Li, Xun Zhou, Xiangnan Kong, and Jun Luo. Curb-gan: Conditional urban traffic estimation through spatio-temporal generative adversarial networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, KDD '20, 2020. 5, 58, 61, 74, 76, 83, 87, 88, 90, 99, 101, 110

[3] Yingxue Zhang, Yanhua Li, Xun Zhou, Zhenming Liu, and Jun Luo. C^3-GAN: Complex-condition-controlled urban traffic estimation through generative adversarial networks. In *2021 IEEE International Conference on Data Mining (ICDM)*, 2021. 5, 88, 110

[4] World urbanization prospects 2014, 2014. 7

[5] Jake Hooker. *Beijing Announces Traffic Plan for Olympics*. The New York Times, 2008. 8

[6] Haowei Su and Shu Yu. Hybrid GA based online support vector machine model for short-term traffic flow forecasting. In *APPT*, pages 743–752, 2007. 8

[7] Ryan Herring, Aude Hofleitner, Pieter Abbeel, and Alexandre M. Bayen. Estimating arterial traffic conditions using sparse probe data. In *ITSC*, pages 929–936, 2010. 8, 34, 57, 83, 88

[8] Pablo Samuel Castro, Daqing Zhang, and Shijian Li. Urban traffic modelling and prediction using large scale taxi GPS traces. In *Pervasive*, pages 57–72, 2012. 8, 34, 57, 83, 88

[9] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. T-drive: driving directions based on taxi trajectories. In *GIS*, pages 99–108, 2010. 8, 34

[10] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. Driving with knowledge from the physical world. In *KDD*, pages 316–324, 2011. 8, 34

[11] Benhamza Karima, Salah Ellagoune, Hamid Seridi, and Herman Akdag. Agent-based modeling for traffic simulation. June 2019. 8, 38

[12] John Taplin. Simulation models of traffic flow. 08 2008. 8, 38

[13] Anika Singh Lemar. Zoning as Taxidermy: Neighborhood Conservation Districts and the Regulation of Aesthetics. *Indiana Law Journal*, 90:1525–1590, 09 2015. 9

[14] Naoto Mukai and Naoto Yoden. Taxi demand forecasting based on taxi probe data by neural network. In *IIMSS*, pages 589–597, 2012. 11, 50, 74, 99, 145

[15] Eric J. Gonzales, Ci (Jesse) Yang, Ender Faruk Morgul, and Kaan Ozbay. Modeling taxi demand with gps data from taxis and transit. Technical report, Mineta National Transit Research Consortium, 2014. 11, 50, 74, 87, 99, 145

[16] Michael G. McNally. *The Four-Step Model*, chapter 3, pages 35–53. 12, 32

[17] Xianyuan Zhan, Yu Zheng, Xiuwen Yi, and Satish Ukkusuri. Citywide traffic volume estimation using trajectory data. *TKDE*, 29(2):272–285, 2017. 14, 38, 60, 88, 137

[18] Xinyue Liu, Xiangnan Kong, and Yanhua Li. Collective traffic prediction with partially observed traffic history using location-base social media. In *CIKM*, pages 2179–2184, 2016. 14, 38, 60, 88

[19] Ermal Toto, Elke A. Rundensteiner, Yanhua Li, Richard Jordan, Mariya Ishutkina, Kajal Claypool, Jun Luo, and Fan Zhang. Pulse: A real time system for crowd flow prediction at metropolitan subway stations. In *ECMLPKDD*, pages 112–128, 2016. 14, 38, 61, 88, 137

[20] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017. 14

[21] Hao Wu, Ziyang Chen, Weiwei Sun, Baihua Zheng, and Wei Wang. Modeling trajectories with recurrent neural networks. In *IJCAI*, pages 341–350, 2017. 15

[22] Chao Huang, Junbo Zhang, Yu Zheng, and Nitesh V Chawla. DeepCrime: Attentive hierarchical recurrent networks for crime prediction. In *CIKM*, pages 1423–1432, 2018. 15, 39

[23] Yexin Li, Yu Zheng, and Qiang Yang. Dynamic bike reposition: A spatio-temporal reinforcement learning approach. In *KDD*, pages 1724–1733, 2018. 15, 39

[24] Zhuoning Yuan, Xun Zhou, and Tianbao Yang. Hetero-convlstm: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 984–992, 2018. 15, 132, 137

[25] Yanhua Li, Moritz Steiner, Jie Bao, Liming Wang, , and Ting Zhu. Region sampling and estimation of geosocial data with dynamic range calibration. In *ICDE*, pages 1096–1107, 2014. 15

[26] Yanhua Li, Jun Luo, Chi-Yin Chow, Kam-Lam Chan, Ye Ding, and Fan Zhang. Growing the charging station network for electric vehicles with trajectory data analytics. In *ICDE*, pages 1376–1387, 2015. 15

[27] Bing Zhu and Xin Xu. Urban principal traffic flow analysis based on taxi trajectories mining. In *ICSI*, pages 172–181, 2015. 16

[28] Jingyi Guo, Xianghua Li, Zili Zhang, and Junwei Zhang. Traffic flow fluctuation analysis based on beijing taxi gps data. In *KSEM*, pages 452–464, 2018. 16

[29] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, pages 2672–2680. 2014. 17, 62, 63, 88, 89, 116

[30] Jon Gauthier. Conditional generative adversarial nets for convolutional face generation. 2015. 17, 28

[31] Jan Hauke and Tomasz Kossowski. Comparison of values of pearson's and spearman's correlation coefficients on the same sets of data. *Quaestiones Geographicae*, 30(2):87 – 93, 2011. 18

[32] Waldo R Tobler. A computer movie simulating urban growth in the detroit region. *Economic geography*, 46(sup1):234–240, 1970. 19, 40

[33] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. In *CoRR abs/1411.1784*, 2014. 23, 28, 39, 45, 61, 62, 75, 89, 101, 116

[34] Arthur Getis. A history of the concept of spatial autocorrelation: A geographer's perspective. *Geographical Analysis*, 40:297–309, 07 2008. 28, 48, 100

[35] Ishteaque Alam, Dewan Md. Farid, and Rosaldo J. F. Rossetti. The prediction of traffic flow with regression analysis. In *IEMIS*, pages 661–671, 2019. 28, 146

[36] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. Urban computing: concepts, methodologies, and applications. *TIST*, 2014. 33, 130

[37] Dina Al-Shibeeb. *Vaughan council rejects Sports Village expansion after 4-year debate*. YorkRegion, 2019. 33

[38] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, Fei-Yue Wang, et al. Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 2015. 35, 57, 132

[39] Zhuoning Yuan, Xun Zhou, and Tianbao Yang. Hetero-convlstm: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data. 2018. 35, 39, 57

[40] Ali Zonoozi, Jung jae Kim, Xiao-Li Li, and Gao Cong. Convolutional recurrent model for crowd density prediction with recurring periodic patterns. In *IJCAI*, 2018. 35, 39, 61, 88

[41] Haiyang Yu, Zhihai Wu, Shuqin Wang, Yunpeng Wang, and Xiaolei Ma. Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks. *Sensors*, 2017. 35, 39, 61, 88, 132

[42] Zhiyong Cui, Ruimin Ke, and YinhaiWang. Deep stacked bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction. In *6th International Workshop on Urban Computing*, 2017. 35, 132

[43] Zipei Fan, Xuan Song, Ryosuke Shibasaki, and Ryutaro Adachi. Citymomentum: An online approach for crowd behavior prediction at a citywide level. In *ACM UbiComp*, 2015. 38, 88

[44] Xuan Song, Quanshi Zhang, Yoshihide Sekimoto, and Ryosuke Shibasaki. Prediction of human emergency behavior and their mobility following large-scale disaster. In *SIGKDD*, 2014. 38, 88

[45] Junbo Zhang, Yu Zheng, and Dekang Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. *CoRR*, 2016. 39, 61, 88, 137

[46] Huaxiu Yao, Fei Wu, Jintao ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, and Jieping Ye. Deep multi-view spatial-temporal network for taxi demand prediction. 2018. 39, 61, 88

[47] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, 2014. 39, 61, 88

[48] Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S. Yu. Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. In *NIPS*, 2017. 39, 88

[49] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. When will you arrive? estimating travel time based on deep neural networks. In *AAAI*, 2018. 39, 61, 88

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, 2017. 40, 42, 45, 48, 123

[51] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 2015. 48, 57, 61, 88, 113, 118, 119

[52] Olof Mogren. C-RNN-GAN: continuous recurrent neural networks with adversarial training. *CoRR*, 2016. 48, 61, 62, 116, 146

[53] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, 2018. 48

[54] Jiachen Zhao, Fang Deng, Yeyun Cai, and Jie Chen. Long short-term memory - fully connected (lstm-fc) neural network for pm2.5 concentration prediction. *Chemosphere*, 2019. 48, 146

[55] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. *CoRR*, 2014. 48

[56] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 50, 77, 102, 120, 125, 143, 148

[57] R. A. Anand, L. Vanajakshi, and S. C. Subramanian. Traffic density estimation under heterogeneous traffic conditions using data fusion. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 31–36, 2011. 57, 83, 88

[58] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 1997. 61, 88, 124

[59] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 61

[60] Y. Choi, M. Choi, M. Kim, J. Ha, S. Kim, and J. Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 61

[61] Ayush Jaiswal, Wael AbdAlmageed, Yue Wu, and Premkumar Natarajan. Bidirectional conditional generative adversarial networks, 2018. 61

[62] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *CoRR*, 2016. 62, 63, 75, 117

[63] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013. 67

[64] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010. 68

[65] V. Mnih, K. Kavukcuoglu, D. Silver, and et al. Human-level control through deep reinforcement learning. *Nature*, 2015. 70

[66] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 75, 91, 101

[67] P. Chonwiharnphan, P. Thienprapasith, and E. Chuangsuwanich. Generating realistic users using generative adversarial network with recommendation-based embedding. *IEEE Access*, 2020. 75

[68] Guim Perarnau, Joost van de Weijer, Bogdan Raducanu, and Jose M. Álvarez. Invertible conditional gans for image editing. *CoRR*, 2016. 75

[69] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *ICLR*, 2017. 75

[70] Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 76, 101

[71] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. 77

[72] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, 2016. 81

[73] Leye Wang, Bin Guo, and Qiang Yang. Smart city development with urban transfer learning. *Computer*, 51(12):32–41, 2018. 83, 88

[74] Leye Wang, Xu Geng, Xiaojuan Ma, Feng Liu, and Qiang Yang. Cross-city transfer learning for deep spatio-temporal prediction. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 1893–1899. International Joint Conferences on Artificial Intelligence Organization, 7 2019. 83, 88, 101

[75] Tanwi Mallick, Prasanna Balaprakash, Eric Rask, and Jane Macfarlane. Transfer learning with graph neural networks for short-term highway traffic forecasting. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 10367–10374, 2021. 83, 88

[76] Chuanting Zhang, Haixia Zhang, Jingping Qiao, Dongfeng Yuan, and Minggao Zhang. Deep transfer learning for intelligent cellular traffic prediction based on cross-domain big data. *IEEE Journal on Selected Areas in Communications*, 37(6):1389–1401, 2019. 83, 88

[77] Ying Wei, Yu Zheng, and Qiang Yang. Transfer knowledge between cities. KDD '16. Association for Computing Machinery, 2016. 84, 88

[78] Tianfu He, Jie Bao, Ruiyuan Li, Sijie Ruan, Yanhua Li, Li Song, Hui He, and Yu Zheng. What is the human mobility in a new city: Transfer mobility knowledge across cities. In *Proceedings of The Web Conference 2020*, WWW '20, page 1355–1365, New York, NY, USA, 2020. Association for Computing Machinery. 84, 89

[79] Huaxiu Yao, Yiding Liu, Ying Wei, Xianfeng Tang, and Zhenhui Li. Learning from multiple cities: A meta-learning approach for spatial-temporal prediction. In *The World Wide Web Conference*, WWW '19, page 2181–2191, New York, NY, USA, 2019. Association for Computing Machinery. 84, 89, 101

[80] Leye Wang, Xu Geng, Jintao Ke, Chen Peng, Xiaojuan Ma, Daqing Zhang, and Qiang Yang. Ridesourcing car detection by transfer learning. *CoRR*, 2017. 88

[81] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. 92

[82] Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. 03 2018. 94, 95, 113, 137

[83] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of COMPSTAT'2010*, pages 177–186. Physica-Verlag HD, 2010. 94

[84] Louis Clouâtre and Marc Demers. FIGR: few-shot image generation with reptile. *CoRR*, 2019. 95

[85] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, page 1126–1135, 2017. 101, 113, 137

[86] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. 107, 112, 113, 115, 123

[87] Menghai Pan, Weixiao Huang, Yanhua Li, Xun Zhou, and Jun Luo. Xgail: Explainable generative adversarial imitation learning for explainable human decision analysis. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD '20, page 1334–1343, 2020. 107, 115

[88] Xin Zhang, Yanhua Li, Xun Zhou, and Jun Luo. cgail: Conditional generative adversarial imitation learning—an application in taxi drivers' strategy learning. *IEEE Transactions on Big Data*, pages 1–1, 2020. 107, 112, 115, 124

[89] Xin Zhang, Yanhua Li, Xun Zhou, Ziming Zhang, and Jun Luo. Trajgail: Trajectory generative adversarial imitation learning for long-term decision analysis. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 801–810, 2020. 107, 112, 116, 123, 124

[90] Yunzhu Li, Jiaming Song, and Stefano Ermon. Infogail: Interpretable imitation learning from visual demonstrations. In *Proceedings of the 31st International Con-*

ference on Neural Information Processing Systems, NIPS'17, page 3815–3825. Curran Associates Inc., 2017. 108, 124

[91] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 357–368. PMLR, 13–15 Nov 2017. 108, 113, 124

[92] Dean A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991. 112

[93] Stephane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 661–668. PMLR, 13–15 May 2010. 112

[94] Jonathan Ho, Jayesh K. Gupta, and Stefano Ermon. Model-free imitation learning with policy optimization. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 2760–2769, 2016. 112

[95] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. ICML '04, page 1. Association for Computing Machinery, 2004. 112

[96] Alex Kuefler, Jeremy Morton, Tim Wheeler, and Mykel Kochenderfer. Imitating driver behavior with generative adversarial networks. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 204–211, 2017. 112

[97] James Harrison, Apoorva Sharma, Chelsea Finn, and Marco Pavone. Continuous

meta-learning without tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 17571–17581. Curran Associates, Inc., 2020. 113, 137

[98] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1842–1850. JMLR.org, 2016. 113

[99] Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables, 2019. 113

[100] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2554–2563. PMLR, 06–11 Aug 2017. 113

[101] Yan Duan, Marcin Andrychowicz, Bradly C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning, 2017. 113

[102] Tianhe Yu, Pieter Abbeel, Sergey Levine, and Chelsea Finn. One-shot hierarchical imitation learning of compound visuomotor tasks. *CoRR*, abs/1810.11043, 2018. 113

[103] Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *CoRR*, abs/1802.01557, 2018. 113

[104] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley amp; Sons, Inc., USA, 1st edition, 1994. 114

[105] Rick Durrett. *Probability: Theory and Examples*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 4 edition, 2010. 114

[106] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, page 1433–1438. AAAI Press, 2008. 115

[107] Ben Poole, Sherjil Ozair, Aaron van den Oord, Alexander A. Alemi, and George Tucker. On variational bounds of mutual information, 2019. 117

[108] Jan Gläscher, Nathaniel Daw, Peter Dayan, and John P. O'Doherty. States versus rewards: Dissociable neural prediction error signals underlying model-based and model-free reinforcement learning. *Neuron*, 66(4):585–595, 2010. 119

[109] Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. Large-margin softmax loss for convolutional neural networks, 2017. 119

[110] Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *Proceedings of the International Workshop on Artificial Neural Networks: From Natural to Artificial Neural Computation*, IWANN '96, page 195–201, Berlin, Heidelberg, 1995. Springer-Verlag. 119

[111] B.L. Kalman and S.C. Kwasny. Why tanh: choosing a sigmoidal function. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 4, pages 578–581 vol.4, 1992. 119

[112] Lantao Yu, Tianhe Yu, Chelsea Finn, and Stefano Ermon. Meta-inverse reinforcement learning with probabilistic context variables. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 120

[113] John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. Trust region policy optimization. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 1889–1897. JMLR.org, 2015. 120, 121

[114] Manoel Castro-Neto, Young-Seon Jeong, Myong-Kee Jeong, and Lee Han. Online-svr for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert Systems with Applications*, 36:6164–6173, 2009. 131, 137

[115] Yuxing Sun, Biao Leng, and Guan Wei. A novel wavelet-svm short-time passenger flow prediction in beijing subway system. *Neurocomputing*, 166, 04 2015. 131, 137

[116] Yuliang Cong, Jianwei Wang, and Xiaolei Li. Traffic flow forecasting by a least squares support vector machine with a fruit fly optimization algorithm. *Procedia Engineering*, 137:59 – 68, 2016. 131, 137

[117] Xianglong Luo, Liyao Niu, and Shengrui Zhang. An algorithm for traffic flow prediction based on improved sarima and ga. *KSCE Journal of Civil Engineering*, 22:1–9, 05 2018. 131, 137

[118] Zheyi Pan, Yuxuan Liang, Weifeng Wang, Yong Yu, Yu Zheng, and Junbo Zhang. Urban traffic prediction from spatio-temporal data using deep meta learning. In *KDD*, 05 2019. 132, 138

[119] Huaxiu Yao, Yiding Liu, Ying Wei, Xianfeng Tang, and Zhenhui Li. Learning from multiple cities: A meta-learning approach for spatial-temporal prediction. In *The World Wide Web Conference*, page 2181–2191, 2019. 132, 138, 146

[120] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural

attentive meta-learner. In *International Conference on Learning Representations*, 2018. 137, 146

[121] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. 141

[122] M. Tan, S. C. Wong, J. Xu, Z. Guan, and P. Zhang. An aggregation approach to short-term traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, pages 60–69, 2009. 146

[123] Chelsea Finn. Bayesian meta-learning. `https://cs330.stanford.edu/slides/cs330_bayesian_metalearning.pdf`, 2019. [Online]. 146